SUMINISTROS Y APLICACIONES INDUSTRIALES S.A.

INFORME MEJORA:

IMPLEMENTACIÓN TIENDA WEB ONLINE

Proyecto realizado por

Aguayo Orozco Álvaro, Manzano Montenegro José Julián, Ruiz Porro Jorge.

Enlace a ProjectSII:

https://projetsii.informatica.us.es/projects/c72gnmglfp92wng6ryn

Índice general

1. Introducción al problema

La compañía privada SAIND (Suministros y Aplicaciones Industriales) solicita implementar una tienda online en su página web para reemplazar la que ya tienen.

Actualmente, el servicio que disponen no está en su totalidad informatizado. Los clientes deben de realizar una llamada para encargar cualquier tipo de servicio, el pago debe realizarse una vez el comprador reciba el producto, los datos de compra y reparaciones los debe realizar una persona a mano para luego pasarlos a ordenador.

Una vez finalizada la implementación, se espera conseguir ampliar el número de clientes, agilizar los trámites de pago por parte de estos y tener una lista con sus datos, una mejor visión de los servicios que se venden en cada línea de producto, poder modificar el catálogo disponible ya sea añadiendo nuevos productos o modificando los existentes y comprobar el estado de los pedidos para revisar que no haya ninguno con problemas técnicos.

2. Glosario de Términos

Alquiler – Servicio a disposición de los clientes con una amplia y variada gama de equipos en régimen de alquiler.

Arco eléctrico – Se denomina arco eléctrico a la descarga eléctrica entre dos electrodos sometidos a una diferencia de potencial.

Cliente – Persona que compra en una tienda, o utiliza los servicios de un profesional o empresa.

Consumibles – Productos de uno o varios usos, pero que en definitiva se consumen.

Electrodos – Metal de aporte en la soldadura.

Electrodo revestido – Electrodo con revestimiento que se descompone en forma de gases y escoria líquida con el calor, escora que flota sobre el baño de gases y luego se solidifica.

Empresa – Unidad de organización dedicada a actividades industriales, mercantiles o de prestación de servicios con fines lucrativos.

Escoria – Capa que se forma sobre la soldadura una vez que se han fusionado los metales.

Gas protector/reactor – En la soldadura por arco los metales pueden reaccionar químicamente con elementos presentes en el aire como el oxígeno y el nitrógeno, destruyendo así la resistencia y dureza de la unión. Por eso se utiliza un gas protector que minimice el contacto del metal fundido con el aire.
También se puede usar un gas reactor que participe activamente en la soldadura.

Guantes – Guantes especiales para soldar. Un buen guante para soldar en las diferentes técnicas requiere pieles nobles, hilos resistentes a corte y térmicos, en definitiva, necesitan mucha resistencia y ser muy absorbentes al calor.

Hilos – Son el electrodo en la soldadura, el metal de aporte. Pueden ser de distintos metales y de distintos grosores, mientras más grosor mayor es la incidencia en el metal.

Investigación y desarrollo – Departamento encargado del desarrollo de nuevos equipos y sistemas automáticos de soldadura.

Manómetros – Es un instrumento que mide la presión en un medio. Hay diferentes tipos, como de oxígeno o argón, y están diseñados para medir la presión en un medio con altas concentraciones de esos gases.

Pantallas – Máscara para proteger la cara mientras se está soldando, son electrónicas y se puede ajustar la cantidad de luz que se puede ver a través de ellas para no dañar los ojos.

Línea de producción – Todos los servicios que ofrece la empresa a sus clientes. A saber:

- **Soldaduras:**
  - MIG-MAG
  - TIG
  - MMA
  - Plasma
- Alquiler.
- Repuestos.
- Servicio técnico.
- Servicio mecanizado.
- Piping.
- Viradores.

Part Number – Es el número de referencia de la pieza.

Piping – Equipos y materiales para el trabajo con tubos.

Producto – Objeto que se ofrece en un mercado con intención de satisfacer aquello que necesita o desea un consumidor.

Repuestos – Piezas destinadas a sustituir a otras de la misma clase cuando esta se gasta o se estropea.

Servicio técnico – Personal técnico que se encarga de la puesta en marcha, el mantenimiento y la reparación de los equipos de soldadura suministrados.

Soldadura – Proceso de fijación en donde se realiza la unión de dos o más piezas de un material (generalmente metales o termoplásticos).

Soldadura por arco – Se denomina soldadura por arco a la técnica que se basa en generar un arco eléctrico entre el electrodo y el material base, derritiendo los metales en el punto de soldadura.

Soldadura MIG/MAG – La soldadura MIG/MAG es una soldadura por arco bajo gas protector/reactor con electrodo consumible. Dependiendo del gas inyectado será MIG (Metal Inert Gas) o MAG (Metal Active Gas) siendo el primero un gas que simplemente protege y el segundo un gas que actúa en la reacción.

Soldadura MMA – Soldadura por arco con electrodo revestido.

Soldadura TIG – Sistema de soldadura por arco que utiliza el intenso calor de un arco eléctrico generado entre un electrodo de tungsteno no consumible y la pieza a soldar. Se utiliza gas protector en la soldadura.

Soldadura por plasma – Utiliza los mismos principios que la soldadura TIG, sin embargo, tanto la densidad energética y la temperatura son mucho mayores ya que hay que alcanzar el estado plasmático. La mayor ventaja de este proceso es que la zona de impacto es dos o tres veces inferior en comparación con la soldadura TIG, por lo que se convierte en óptima para soldar metal de espesores pequeños.

Torchas – La torcha es el aparato que se utiliza propiamente para soldar, es decir, en soldadura por arco, es el aparato donde se genera el arco eléctrico y donde se calienta el electrodo para empezar a soldar.

Tungsteno – También llamado "wolframio", es un metal escaso en la corteza terrestre. Tiene el punto de fusión más elevado de todos los metales por lo que se usa para soldaduras como electrodo no consumible.

Válvulas – Regulan la presión en los tubos que transportan los distintos gases utilizados en la soldadura.

Varillas – Alambre o varilla metálica de aporte, se emplea en soldaduras en las que el electrodo no proporciona suficiente metal de aporte.

Viradores – Aparatos que mediante ruedas permiten el fácil giro de un cuerpo cilíndrico. Son de gran utilidad para la ejecución de soldaduras en tubos y recipientes. Su utilidad variable sin escalonamientos permite según la naturaleza de los trabajos a realizar, utilizar todos los procedimientos de soldadura.

3. Visión General del Sistema.

Al implementar la Tienda Online necesitaremos una base de datos que contenga todos los productos y servicios que ofrecerá la empresa por medio de Internet, esos productos estarán catalogados por categorías y cada uno tendrá su precio, se les aplicará las ofertas que los dueños vean necesarias. Cada producto tendrá todo lo necesario para identificarle, como se explica en la zona **"Líneas de productos"**.

Tanto los portes como las ofertas serán añadidos automáticamente al precio del pedido, pudiendo ser los portes gratuitos. El método de pago deberá ser seguro para los clientes y permitirá varias formas de pago distintas.
También se implementará una base de datos para almacenar los usuarios. Los clientes tendrán acceso a su información exclusivamente, su historial de pedidos y un apartado de seguimiento de sus pedidos, todo ello en un panel de control de usuario.

Para los trabajadores de la empresa se implementarán distintos sistemas. Un método de notificaciones para que el almacén sepa cuando se haya realizado un pedido en la web y otro con la información necesaria sobre los usuarios, como los clientes de la empresa, la cantidad de pedidos por cliente, el estado de estos pedidos, clientes con facturas impagadas, problemas con pedidos y productos con problemas técnicos. Todo ello en una misma interfaz con diferentes tipos de usuarios (Usuario: Cliente, Usuario: Dirección, Usuario: Almacén) con distintos permisos de acceso dentro de la empresa, para que cada trabajador tenga más fácil y rápido acceso a su zona de trabajo en la web.

Tienda online:

Como director de la empresa,

Quiero disponer de la siguiente información:

- Clientes.
- Número y cantidad de pedidos por cliente.
- Estado de los pedidos por cliente.
- Lista de clientes con facturas impagadas.
- Problemas con los pedidos.
- Productos con más problemas técnicos.

Líneas de productos:

Como director de la empresa,

Quiero que cada producto tenga todos sus atributos identificativos y documentación técnica según proceda,

Los equipos de soldadura tendrán su Part number, modelo, una descripción, deberá tener adjunto un PDF con documentación técnica.

Los consumibles deberán tener nombre, diámetro, longitud (según el producto), descripción y tipo.

Los Repuestos deberán llevar su nombre, descripción, y tipo.

Gestión de pedidos:

Como encargado de almacén,

Quiero que cuando se realice un pedido en la web, se notifique en almacén automáticamente.

Tipo de tienda:

Como secretaria de dirección,

Quiero que se puedan añadir o eliminar ofertas en la tienda web con un marco de tiempo concreto, al realizar el pedido quiero que se añadan automáticamente los portes sean gratuitos o no.

Pedidos:

Como cliente,

Quiero que se me notifique cuando mi pedido esté listo para recoger o para su envió.

Pedidos:

Como cliente,

Quiero tener un historial de los pedidos realizados.

Pedidos:

Como cliente,

Quiero que cuando se realice un pedido en la web, se apliquen descuentos y ofertas automáticamente si procede.

Formas de pago:

Como director de la empresa,

Quiero que la forma de pago sea por adelantado y por varias vías de pago.

4. Catálogo de Requisitos.

• Objetivos.

Tienda Online:

Como cliente,

Quiero una plataforma que me permita encargar un producto sin tener que llamar por teléfono a la empresa.

Tienda Online:

Como director de la empresa,

Quiero un servicio que permita la gestión de la compra de un cliente.

Gestión de pedidos:

Como encargado,

Quiero que cuando se realice un pedido en la web, se notifique éste a los empleados de almacén automáticamente.

• Requisitos de información.

RI-01 Información de los pedidos:

Como director de la empresa,

Quiero disponer de la siguiente información:

- Código identificativo del pedido.
- Fecha y hora.
- Estado de los pedidos.
- Datos para envió.

RI-02 Información de las líneas de pedido:

Como director de la empresa,

Quiero que disponer, de los datos de las ofertas:

- Cantidad pedida.
- Precio esperado.

RI-03 Información de los productos:

Como director de la empresa,

Quiero que disponer, almacenar de cada producto, todos sus atributos identificativos y documentación técnica según su tipo:

- Los equipos de soldadura:
  - Imagen.
  - Part number.
  - Modelo.
  - PDF con documentación técnica.
- Los consumibles:
  - Imagen.
  - Nombre.
  - Dimensiones.
  - Descripción.
  - Tipo.
- Los Repuestos:
  - Imagen.
  - Nombre
  - Descripción
  - Tipo.

RI-04 Información de los clientes:

Como director,

Quiero disponer y almacenar los siguientes datos de los clientes:

- Nombre del cliente o empresa.
- Código identificativo del cliente
- DNI o CIF.
- Teléfono de contacto.
- Correo Electrónico.
- Dirección.

RI-05 Información de las ofertas:

Como director de la empresa,

Quiero que disponer, de los datos de las ofertas:

- Código de oferta.
- Fecha de inicio.
- Fecha fin.
- Precio Ofertado.

RI-06 Información Software de gestión:

El software de gestión que usa la empresa es Sage FacturaPlus Elite, si es posible la web deberá agregar el pedido a dicho software, si no fuera posible o hubiera problemas técnicos, un empleado se encargará de comprobar y tramitar los datos de los pedidos.

RI-07 Información de empleados:

Como director,

Quiero disponer de estos datos:

- Nombre.
- ID
- DNI

RI-08 Información de almacén:

Como director,

Quiero disponer de estos datos:

- Nombre.
- Dirección.

RI-09 Información de la factura:

Como director,

La factura deberá contener la siguiente información:

- Forma de pago.
- Código identificativo.
- Precio total.
- Datos del pedido

RI-10 Información del albarán:

Como director,

El albarán deberá contener la siguiente información:

- Código identificativo.
- Datos del pedido

• Reglas de Negocio.

RN-01 Clientes:

Como director,

Quiero que no se puedan eliminar los datos correspondientes a un cliente mientras éste tenga pedidos pendientes.

RN-02 Clientes:

Como director,

Quiero que los pedidos realizados a partir de las 17:00, se queden en espera para el día siguiente.

RN-03 Clientes:

Como director,

Quiero que no le llegue una notificación de pedido al empleado hasta que no se confirme el pago.

RN-04 Clientes:

Como director,

Quiero que antes de realizar él envió de un pedido, se compruebe que el pago no ha sido cancelado.

• Requisitos Funcionales.

RF-01 Tipo de tienda:

Como secretaria de dirección,

Quiero que se puedan añadir o eliminar ofertas en la tienda web con un marco de tiempo concreto, al realizar el pedido quiero que se añadan automáticamente.

RF-02 Pedidos:

Como cliente,

Quiero que se me notifique cuando mi pedido sea confirmado.

RF-03 Historial de pedidos:

Como cliente,

Quiero tener un historial de los mis pedidos, por fecha.

RF-04 Formas de pago:

Como director de la empresa,

Quiero que la forma de pago sea por adelantado, al realizar el pedido, mediante PayPal, transferencia bancaria o tarjeta de crédito.

RF-05 Gestión de pedidos:

Como encargado de almacén,

Quiero recibir una notificación cuando un pedido esté listo para tramitar.

- # Requisitos no Funcionales.

RNF-01: Control de pedidos:

Como director,

Quiero que cuando surja un incidente con un pedido, se le comunique al cliente a ser posible en menos de 24H.

RNF-02: Disponibilidad:

Como director,

Quiero que una disponibilidad de la tienda web los más cercana a 24H/7D.

RNF-03: Compatibilidad:

Como encargado de la web,

Quiero que la tienda sea compatible con la actual web de la empresa.

5. Prueba de aceptación

Escenario 01:

Como director,

Quiero que no se puedan eliminar los datos correspondientes a un cliente mientras éste tenga pedidos pendientes.

Caso de prueba:

- Se registra un cliente nuevo, se pide un listado de clientes y aparece el cliente nuevo.
- Se modifican los datos de un cliente, se pide un listado de clientes y aparece el cliente con los datos modificados.
- Se intenta eliminar un cliente, se comprueba que no tenga pedidos pendientes, si no los tiene se elimina, en caso contrario se conserva.

Escenario 02:

Como director,

Quiero que los pedidos realizados a partir de las 17:00, se queden en espera para el día siguiente.

Prueba de Aceptación:

- Un cliente realiza un pedido antes de las 17:00, se envía una notificación al empleado y se tramita.
- Un cliente realiza un pedido después de las 17:00, se cambia el estado de pedido a "en espera" al día siguiente se comprueba si hay pedidos en espera y se tramitan.

Escenario 03:

Como director,

Quiero que no le llegue una notificación de pedido al empleado hasta que no se confirme el pago.

Prueba de Aceptación:

- El cliente realiza un pedido, se tramita correctamente el pago, se envía una notificación al empleado para tramitar el pedido.
- El cliente realiza un pedido, pero paga por transferencia o hay un problema con el pago, el pedido pasa a estado en espera y no se notifica al empleado hasta que se confirme el pago.

Escenario 04:

Como director,

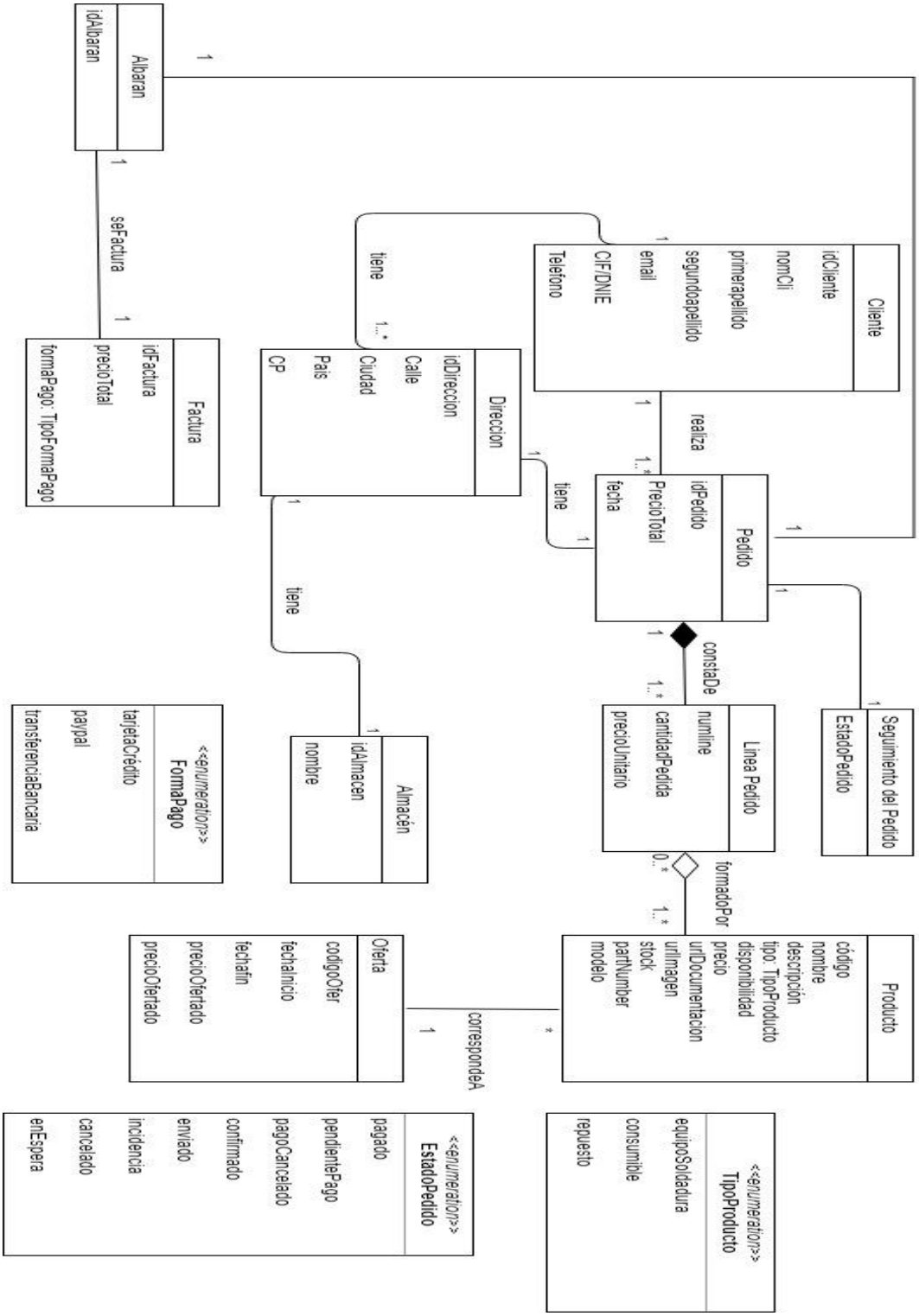
Quiero que antes de realizar él envió de un pedido, se compruebe que el pago no ha sido cancelado.

Prueba de Aceptación:

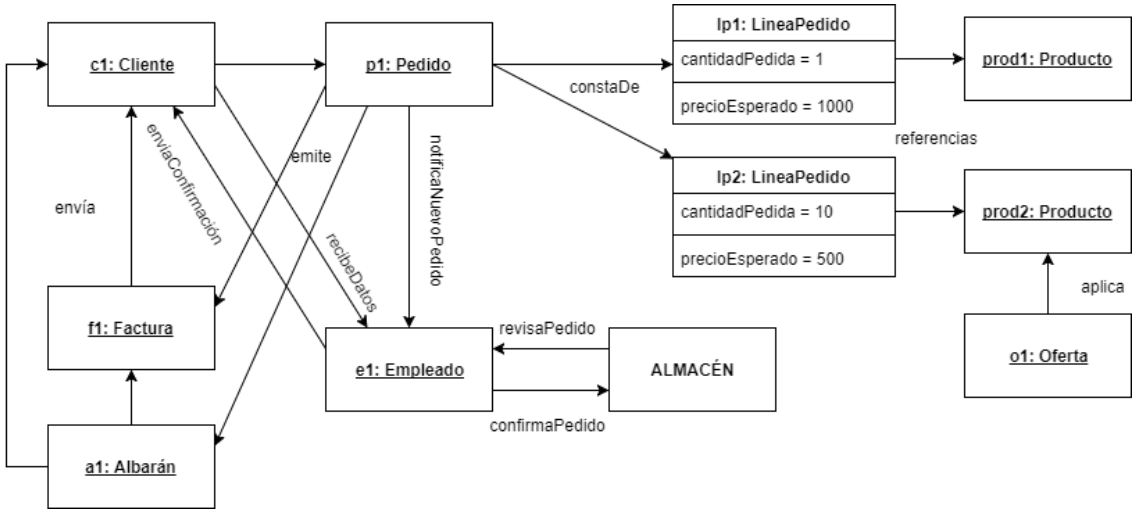
- El cliente realiza un pedido, se comprueba que el pago no ha sido cancelado y se envía.
- El cliente realiza un pedido, el pago se ha cancelado y no se envía.

6. Modelo Conceptual.

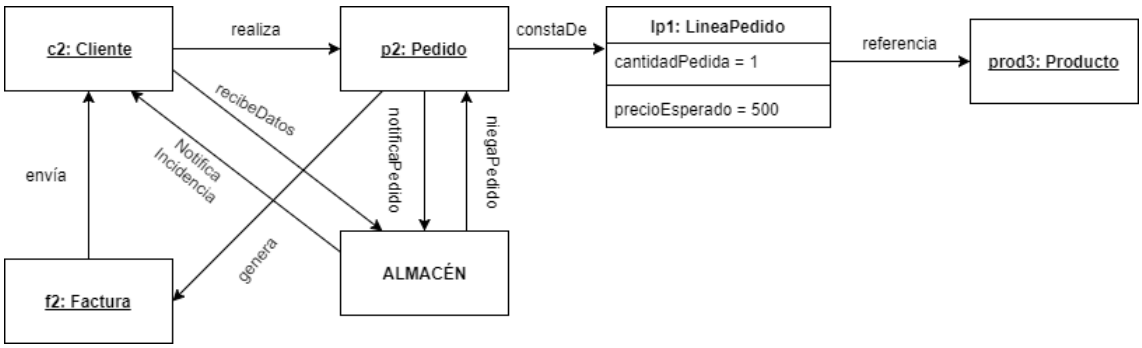
- **Diagrama UML.**



- **Escenario de prueba**



En este primer caso tenemos un pedido realizado satisfactoriamente.
El cliente realiza un pedido que consta de dos productos, uno de ellos con una oferta aplicada, este pedido emite una factura que se le envía al cliente.
Se le notifica al empleado de un nuevo pedido y de los datos del cliente. Si los datos están en orden y que se ha pagado la factura se procede a revisar y confirmar el pedido para su envío.

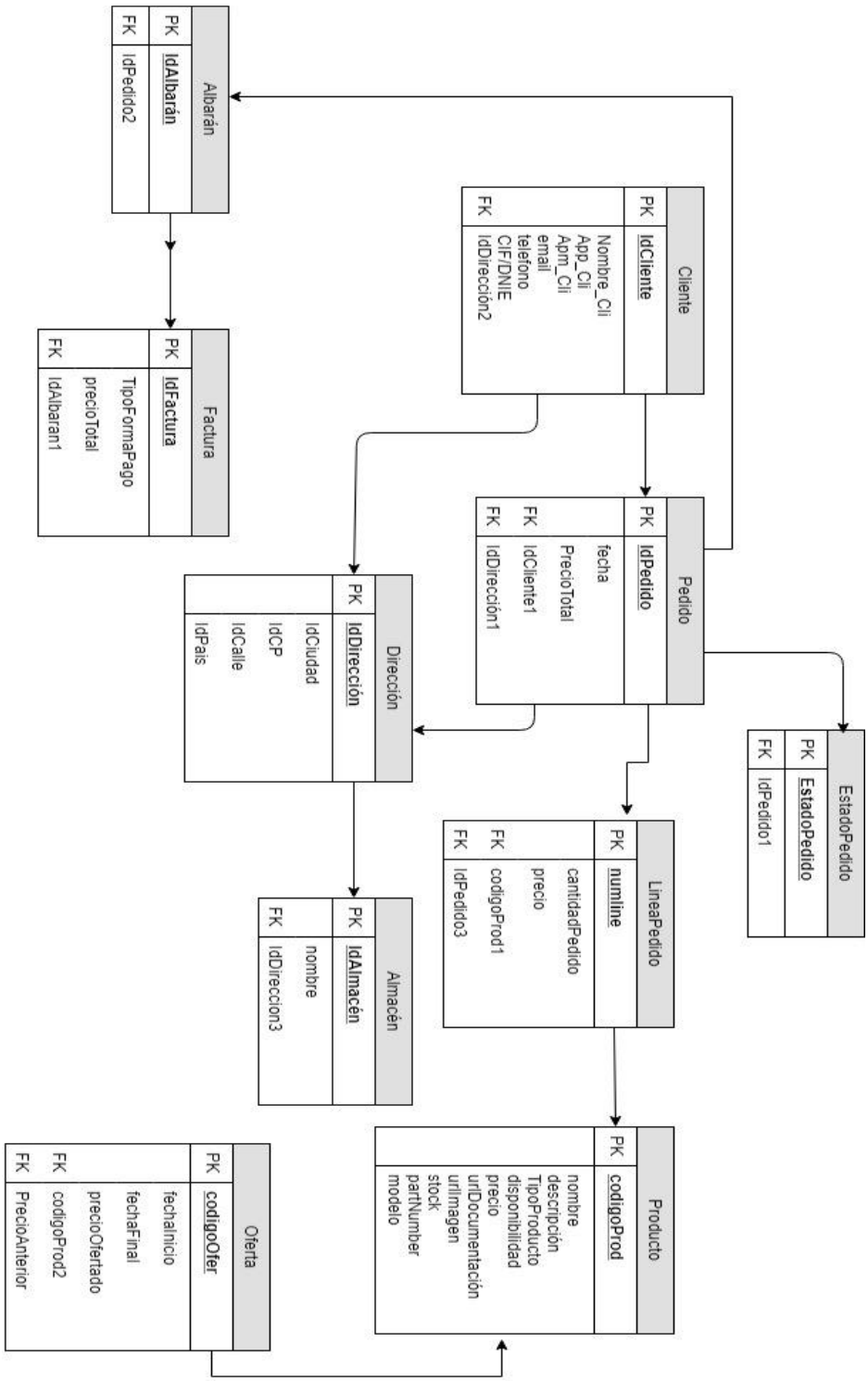


En este segundo caso tenemos a un cliente con una incidencia en el pedido, al intentar realizar el pedido, se han recibido sus datos en el almacén y el sistema Deja el estado del pedido "incidencia" una vez que los empleados tramiten la incidencia el pedido pasara a estado "en espera", posteriormente cuando la incidencia este resuelta el estado del pedido pasara a "cancelado", "confirmado" o "enviado" según proceda.

7. Matrices de trazabilidad

R/C-A	RI-01	RI-02	RI-03	RI-04	RI-05	RI-06	RI-07	RI-08	RI-09	RI-10	RN-01	RN-02	RN-03	RN-04	RF-01	RF-02	RF-03	RF-04	RF-05
Pedidos	✓										✓	✓	✓	✓		✓	✓		✓
Línea de productos		✓																	
Productos			✓												✓				
Clientes				✓							✓		✓	✓		✓	✓		
Ofertas					✓										✓		✓		
Software de gestión						✓					✓	✓			✓				
Empleado							✓				✓		✓	✓	✓	✓			✓
Almacén								✓				✓				✓			
Factura									✓									✓	
Albarán										✓								✓	
notificaEnvio				✓			✓	✓						✓					
genera	✓						✓		✓	✓									
gestiona					✓	✓		✓			✓		✓	✓	✓	✓			
seNotifica	✓			✓															
correspondeA			✓			✓	✓												
formadoPor		✓	✓																
constaDe	✓	✓	✓																
realiza	✓			✓															
seFactura									✓	✓									

8. Modelo relacional en 3FN



9. Modelo tecnológico

Borrado y creado de Tablas.

```
-------------------------------------------------

-- CREACION/ BORRADO DE TABLAS

-------------------------------------------------

-------------------------------------------------

--  BORRADO DE TABLAS

-------------------------------------------------

DROP TABLE FACTURAS;

DROP TABLE ALBARANES;

DROP TABLE ALMACENES;

DROP TABLE LINEAPEDIDOS;

DROP TABLE ESTADOPEDIDOS;

DROP TABLE PEDIDOS;

DROP TABLE CLIENTES;

DROP TABLE OFERTAS;

DROP TABLE PRODUCTOS;

DROP TABLE DIRECCIONES;
```

```
----------------------------------------------

--  CREACION DE TABLAS

----------------------------------------------

CREATE TABLE DIRECCIONES(

ID_DIRECCION VARCHAR(50),

CIUDAD VARCHAR(50),

CP INT,

CALLE VARCHAR(50),

NUMERO INT,

PRIMARY KEY(ID_DIRECCION));


CREATE TABLE PRODUCTOS(

CODIGO VARCHAR(50),

NOMBRE VARCHAR(50),

DESCRIPCION VARCHAR(300),

TIPOPRODUCTO VARCHAR(30) NOT NULL
CHECK(TIPOPRODUCTO
IN('equipoSoldadura','consumible','repuesto')),

DISPONIBILIDAD VARCHAR(20) NOT NULL
CHECK(DISPONIBILIDAD IN('Disponible','No disponible')),

PRECIO FLOAT, URLDOCU VARCHAR(300), URLIMG
VARCHAR(300), STOCK INT,

PARTNUMBER VARCHAR(50), MODELO VARCHAR(50),
PRIMARY KEY(CODIGO));
```

```
CREATE TABLE CLIENTES(IDCLIENTE VARCHAR(50),
NOMBRE_CLI VARCHAR(20), APP_CLI VARCHAR(20),
APM_CLI VARCHAR(20), EMAIL VARCHAR(50), DNI CHAR(9)
UNIQUE, TELEFONO INT, ID_DIRECCION2 VARCHAR(50),
PRIMARY KEY(IDCLIENTE),

FOREIGN KEY(ID_DIRECCION2) REFERENCES
DIRECCIONES(ID_DIRECCION));


CREATE TABLE PEDIDOS( IDPEDIDO VARCHAR(50), FECHA
DATE, PRECIOTOTAL FLOAT,

IDCLIENTE1 VARCHAR(50), ID_DIRECCION1 VARCHAR(50),
PRIMARY KEY(IDPEDIDO),

FOREIGN KEY(IDCLIENTE1) REFERENCES
CLIENTES(IDCLIENTE),

FOREIGN KEY(ID_DIRECCION1) REFERENCES
DIRECCIONES(ID_DIRECCION));

CREATE TABLE ESTADOPEDIDOS(

ESTADOPEDIDO VARCHAR(20) NOT NULL
CHECK(ESTADOPEDIDO IN('Enviado','En proceso',

'Incidencia','Cancelado','Pagado','Pendiente de pago','Pago
cancelado','Confirmado','Enviado','En espera')),

IDPEDIDO1 VARCHAR(50),

PRIMARY KEY(ESTADOPEDIDO),

FOREIGN KEY(IDPEDIDO1) REFERENCES
PEDIDOS(IDPEDIDO));

CREATE TABLE LINEAPEDIDOS(

NUMLINE VARCHAR(50),

CANTIDADPEDIDO INT,
```

```
PRECIO FLOAT,

CODIGO_PROD1 VARCHAR(50),

IDPEDIDO3 VARCHAR(50),

PRIMARY KEY(NUMLINE),

FOREIGN KEY(CODIGO_PROD1) REFERENCES
PRODUCTOS(CODIGO),

FOREIGN KEY(IDPEDIDO3) REFERENCES
PEDIDOS(IDPEDIDO));

CREATE TABLE OFERTAS(

CODIGO VARCHAR(50),

FECHA_INICIO DATE,

FECHA_FIN DATE,

PRECIOOFERTADO FLOAT,

CODIGO_PROD2 VARCHAR(50),

PRECIOANTES FLOAT,

PRIMARY KEY(CODIGO),

FOREIGN KEY(CODIGO_PROD2) REFERENCES
PRODUCTOS(CODIGO));

CREATE TABLE ALMACENES(

IDALMACEN VARCHAR(50),

NOMBRE VARCHAR(50),

ID_DIRECCION3 VARCHAR(50),

PRIMARY KEY(IDALMACEN),

FOREIGN KEY(ID_DIRECCION3) REFERENCES
DIRECCIONES(ID_DIRECCION));
```

```
CREATE TABLE ALBARANES(

IDALBARAN VARCHAR(50),

IDPEDIDO2 VARCHAR(50),

PRIMARY KEY(IDALBARAN),

FOREIGN KEY(IDPEDIDO2) REFERENCES
PEDIDOS(IDPEDIDO));

CREATE TABLE FACTURAS(

IDFACTURA VARCHAR(50),

TIPOFORMAPAGO VARCHAR(30) NOT NULL
CHECK(TIPOFORMAPAGO IN('Tarjeta de
Crédito','PayPal','Transferencia Bancaria')),

PRECIOTOTAL FLOAT,

IDALBARAN1 VARCHAR(50),

PRIMARY KEY(IDFACTURA),

FOREIGN KEY(IDALBARAN1) REFERENCES
ALBARANES(IDALBARAN));
```

Procedimientos, funciones.

```
--Función para calcular precio total del pedido

CREATE OR REPLACE FUNCTION PRECIO_TOTAL

   (INDPEDIDO IN PEDIDOS.IDPEDIDO%TYPE)

   RETURN NUMBER
```

```
IS
   PRECIOFINAL PEDIDOS.PRECIOTOTAL%TYPE;
BEGIN
   SELECT SUM(PRODUCTOS.PRECIO) INTO PRECIOFINAL
   FROM PRODUCTOS, LINEAPEDIDOS
   WHERE LINEAPEDIDOS.ID_PEDIDOX = INDPEDIDO
   AND LINEAPEDIDOS.CODIGO_PRODX =
PRODUCTOS.CODIGO;
   RETURN PRECIOFINAL;
END PRECIO_TOTAL;



--Función para aplicar ofertas
CREATE OR REPLACE FUNCTION APLICA_OFERTA
   (CODIG IN OFERTAS.CODIGO%TYPE)
   RETURN NUMBER
IS
   fechaActual OFERTAS.FECHA_INICIO%TYPE;
   fechaini    OFERTAS.FECHA_INICIO%TYPE;
```

```
fechafin    OFERTAS.FECHA_INICIO%TYPE;
nuevoPrecio OFERTAS.PRECIOOFERTADO%TYPE;
precio OFERTAS.PRECIOANTES%TYPE;
producto    OFERTAS.CODIGO_PROD2%TYPE;
porcentaje  NUMBER;
BEGIN
SELECT TO_CHAR(SysDate,'DD/MM/YYYY') todays_date INTO fechaActual FROM dual;
IF fechaActual>= fechaini AND fechaActual>=fechafin then
SELECT OFERTAS.PRECIOOFERTADO INTO nuevoPrecio FROM OFERTAS;
SELECT OFERTAS.CODIGO_PROD2 INTO producto FROM OFERTAS;
UPDATE PRODUCTOS
SET PRECIO = nuevoPrecio
WHERE CODIGO = producto;
ELSE
Raise_application_error(-20600,'No se puede aplicar la oferta, fuera de fecha');
END IF;
```

```
        SELECT OFERTAS.PRECIOANTES INTO precio FROM
OFERTAS;

        SELECT (precio-nuevoPrecio)*100 INTO porcentaje FROM dual;

        RETURN porcentaje;

    END APLICA_OFERTA;



    create or replace FUNCTION ASSERT_EQUALS (salida
BOOLEAN, salida_esperada BOOLEAN)

    RETURN VARCHAR2

    AS

    BEGIN

    IF(salida = salida_esperada) THEN

    RETURN 'EXITO';

    ELSE

    RETURN 'FALLO';

    END IF;

    END ASSERT_EQUALS;

    ------------------------------------------------

    -- Procedimientos de eliminacion o borrado
```

```
--------------------------------------------------


--------------------------------------------------
--  DDL for Procedure BORRAR_ALBARAN
--------------------------------------------------


create or replace PROCEDURE BORRAR_ALBARAN (
   idborrar IN ALBARANES.IDALBARAN%TYPE
   )
IS idalbaran ALBARANES.IDALBARAN%TYPE;


BEGIN
   DELETE FROM albaranes WHERE IDALBARAN = idborrar;
END BORRAR_ALBARAN;
/
--------------------------------------------------
--  DDL for Procedure BORRAR_ALMACEN
--------------------------------------------------
```

```
create or replace PROCEDURE BORRAR_ALMACEN (
   id_almacendel IN ALMACENES.IDALMACEN%TYPE
   )
IS idalmacen ALMACENES.IDALMACEN%TYPE;

BEGIN
   DELETE FROM almacenes WHERE IDALMACEN =
id_almacendel;
END BORRAR_ALMACEN;
/
--------------------------------------------------
--  DDL for Procedure BORRAR_FACTURA
--------------------------------------------------
create or replace PROCEDURE BORRAR_FACTURA (
   idborrar IN FACTURAS.IDFACTURA%TYPE
   )
IS factura FACTURAS.IDFACTURA%TYPE;

BEGIN
```

```
   DELETE FROM FACTURAS WHERE IDFACTURA = idborrar;
END BORRAR_FACTURA;
/
-----------------------------------------------
-- DDL for Procedure BORRAR_ESTADOPEDIDO
-----------------------------------------------
create or replace PROCEDURE BORRAR_ESTADOPEDIDO (
   idborrar IN ESTADOPEDIDOS.IDPEDIDO1%TYPE
   )
IS
BEGIN
   DELETE FROM ESTADOPEDIDOS WHERE IDPEDIDO1 =
idborrar;
END BORRAR_ESTADOPEDIDO;
/
-----------------------------------------------
-- DDL for Procedure ELIMINA_DIRECCION
-----------------------------------------------
create or replace PROCEDURE ELIMINA_DIRECCION(
```

```
      id_direccionborrar IN
DIRECCIONES.ID_DIRECCION%TYPE

)

IS id_direccion direcciones.ID_DIRECCION%TYPE;



BEGIN

   DELETE FROM direcciones WHERE id_direccion =
id_direccionborrar;

END ELIMINA_DIRECCION;

/

-----------------------------------------------

--  DDL for Procedure ELIMINA_PRODUCTO

-----------------------------------------------

create or replace PROCEDURE ELIMINA_PRODUCTO (

   codigoborrar IN productos.CODIGO%TYPE

)

IS codigo productos.CODIGO%TYPE;



BEGIN
```

```
   DELETE FROM productos WHERE codigo = codigoborrar;

END ELIMINA_PRODUCTO;

/

--------------------------------------------------

--  DDL for Procedure ELIMINAR_PEDIDO

--------------------------------------------------

create or replace PROCEDURE ELIMINA_PEDIDO (

   codigoborrar IN PEDIDOS.IDPEDIDO%TYPE

)

IS codigo PEDIDOS.IDPEDIDO%TYPE;


BEGIN

   DELETE FROM PEDIDOS WHERE IDPEDIDO = codigoborrar;

   BORRAR_ESTADOPEDIDO(codigoborrar);

   DELETE FROM LINEAPEDIDOS WHERE IDPEDIDO3 =
codigoborrar;

END ELIMINA_PEDIDO;

/

--------------------------------------------------
```

```
--  DDL for Procedure ELIMINAR_LINEAPEDIDO
------------------------------------------------
create or replace PROCEDURE ELIMINA_LINEAPEDIDO (
      num IN LINEAPEDIDOS.NUMLINE%TYPE
   )
IS NUMLINE LINEAPEDIDOS.NUMLINE%TYPE;
   BEGIN
   DELETE FROM LINEAPEDIDOS WHERE NUMLINE = num;
END ELIMINA_LINEAPEDIDO;
/
------------------------------------------------
--  DDL for Procedure ELIMINAR_OFERTA
------------------------------------------------
create or replace PROCEDURE ELIMINA_OFERTA (
      id_ofertaborrar IN OFERTAS.CODIGO%TYPE
)
IS id_oferta direcciones.ID_DIRECCION%TYPE;
   restaurarprecio PRODUCTOS.PRECIO%TYPE;
   idproducto OFERTAS.CODIGO_PROD2%TYPE;
```

```
BEGIN

   SELECT OFERTAS.PRECIOANTES INTO restaurarprecio
FROM OFERTAS

   WHERE CODIGO = id_ofertaborrar;

   SELECT OFERTAS.CODIGO_PROD2 INTO idproducto FROM
OFERTAS

   WHERE CODIGO = id_ofertaborrar;

   UPDATE PRODUCTOS

   SET PRECIO = restaurarprecio

   WHERE PRODUCTOS.CODIGO = idproducto;

   DELETE FROM OFERTAS WHERE id_oferta = id_ofertaborrar;

END ELIMINA_OFERTA;

/

-----------------------------------------------

--  DDL for Procedure ELIMINAR_CLIENTE

-----------------------------------------------

create or replace PROCEDURE ELIMINAR_CLIENTE (

   id_clienteDEL IN CLIENTES.IDCLIENTE%TYPE

   )
```

```
IS id_cliente CLIENTES.IDCLIENTE%TYPE;


BEGIN
    DELETE FROM clientes WHERE IDCLIENTE = id_clienteDEL;
END ELIMINAR_CLIENTE;
/
-----------------------------------------------


-----------------------------------------------
-- Procedimientos de Creacion
-----------------------------------------------


-----------------------------------------------
--  DDL for Procedure CREAR_ESTADOPEDIDO
-----------------------------------------------
create or replace PROCEDURE CREAR_ESTADOPEDIDO (
    estado IN ESTADOPEDIDOS.ESTADOPEDIDO%TYPE,
    pedido IN ESTADOPEDIDOS.IDPEDIDO1%TYPE
  )
```

```
IS codigo ESTADOPEDIDOS.ESTADOPEDIDO%TYPE;
  BEGIN
     INSERT INTO ESTADOPEDIDOS (ESTADOPEDIDO,
IDPEDIDO1)
     VALUES(estado, pedido);
END CREAR_ESTADOPEDIDO;
/
-----------------------------------------------
--  DDL for Procedure CREAR_ALBARAN
-----------------------------------------------
create or replace PROCEDURE CREAR_ALBARAN (
     id_pedido IN ALBARANES.IDPEDIDO2%TYPE
   )
IS id_albaran ALBARANES.IDALBARAN%TYPE;
   BEGIN
     SELECT sec_albaran.NEXTVAL INTO id_albaran FROM dual;
     INSERT INTO albaranes (IDALBARAN, IDPEDIDO2)
     VALUES(id_albaran, id_pedido);
END CREAR_ALBARAN;
```

```
/

--------------------------------------------------

--  DDL for Procedure CREAR_ALMACEN

--------------------------------------------------

create or replace PROCEDURE CREAR_ALMACEN (

   nombreal IN ALMACENES.NOMBRE%TYPE,

   id_direccional IN ALMACENES.ID_DIRECCION3%TYPE

   )

IS idalmacen ALMACENES.IDALMACEN%TYPE;


BEGIN

      SELECT sec_almacen.NEXTVAL INTO idalmacen FROM dual;

      INSERT INTO almacenes (IDALMACEN, NOMBRE,
ID_DIRECCION3)

      VALUES(idalmacen, nombreal, id_direccional);

END CREAR_ALMACEN;

/

--------------------------------------------------

--  DDL for Procedure CREAR_FACTURA
```

```
-------------------------------------------------

create or replace PROCEDURE CREAR_FACTURA (

    albaran IN FACTURAS.IDALBARAN1%TYPE,

    pago IN FACTURAS.TIPOFORMAPAGO%TYPE

  )

IS factura FACTURAS.IDFACTURA%TYPE;

  precio PEDIDOS.PRECIOTOTAL%TYPE;

  BEGIN

    SELECT SEC_FACTURA.NEXTVAL INTO factura FROM
dual;

    SELECT PEDIDOS.PRECIOTOTAL INTO precio FROM
ALBARANES,PEDIDOS where ALBARANES.IDPEDIDO2 =
PEDIDOS.IDPEDIDO;

    INSERT INTO FACTURAS (IDFACTURA,
TIPOFORMAPAGO,PRECIOTOTAL,IDALBARAN1)

    VALUES(factura, pago, precio,albaran);

END CREAR_FACTURA;

/


-------------------------------------------------
```

```
-- DDL for Procedure CREAR_CLIENTE

----------------------------------------------------

create or replace PROCEDURE CREAR_CLIENTE (

  nombreCLI IN CLIENTES.NOMBRE_CLI%TYPE,

  app_cli IN CLIENTES.APP_CLI%TYPE,

  apm_cli IN CLIENTES.APM_CLI%TYPE,

  email_cli IN CLIENTES.EMAIL%TYPE,

  dni_cli IN CLIENTES.DNI%TYPE,

  tel_cli IN CLIENTES.TELEFONO%TYPE,

  id_direccion_CLI IN CLIENTES.ID_DIRECCION2%TYPE

  )

IS id_cliente CLIENTES.IDCLIENTE%TYPE;



BEGIN

    SELECT sec_cliente.NEXTVAL INTO id_cliente FROM dual;

    INSERT INTO clientes (IDCLIENTE, NOMBRE_CLI,
APP_CLI, APM_CLI, EMAIL, DNI, TELEFONO,
ID_DIRECCION2)

    VALUES(id_cliente, nombreCLI, app_cli, apm_cli, email_cli,
dni_cli, tel_cli, id_direccion_CLI);
```

```
END CREAR_CLIENTE;

/

--------------------------------------------------

--  DDL for Procedure CREAR_OFERTA

--------------------------------------------------

create or replace PROCEDURE CREAR_OFERTA (

      fechaini IN OFERTAS.FECHA_INICIO%TYPE,

      fechafin IN OFERTAS.FECHA_FIN%TYPE,

      precioDes IN OFERTAS.PRECIOOFERTADO%TYPE,

      codigoPro IN OFERTAS.CODIGO%TYPE

   )

IS idOferta OFERTAS.CODIGO%TYPE;

   precioanterior PRODUCTOS.PRECIO%TYPE;

   BEGIN

      SELECT SEC_OFERTA.NEXTVAL INTO idOferta FROM dual;

      SELECT PRODUCTOS.PRECIO INTO precioanterior FROM
PRODUCTOS

      WHERE PRODUCTOS.CODIGO = codigoPro;
```

```
    INSERT INTO OFERTAS (CODIGO,
FECHA_INICIO,FECHA_FIN,PRECIOOFERTADO,CODIGO_PRO
D2,PRECIOANTES)

    VALUES(idOferta,
fechaini,fechafin,precioDes,codigoPro,precioanterior);

END CREAR_OFERTA;

/

----------------------------------------------------

--  DDL for Procedure CREAR_PEDIDO

----------------------------------------------------

create or replace PROCEDURE CREAR_PEDIDO (

    subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,

    cliente IN PEDIDOS.IDCLIENTE1%TYPE,

    direccion IN PEDIDOS.ID_DIRECCION1%TYPE

  )

AS codigo PEDIDOS.IDPEDIDO%TYPE;

  fecha PEDIDOS.FECHA%TYPE;

  BEGIN

    SELECT SEC_PEDIDO.NEXTVAL INTO codigo FROM dual;
```

```
    SELECT TO_CHAR(SysDate,'DD/MM/YYYY HH24:MI:SS')
todays_date INTO fecha FROM dual;

    INSERT INTO PEDIDOS (IDPEDIDO, FECHA,
PRECIOTOTAL, IDCLIENTE1, ID_DIRECCION1)

    VALUES(codigo, fecha, subtotal, cliente, direccion);

    CREAR_ESTADOPEDIDO('En proceso', codigo);

END CREAR_PEDIDO;

/

-----------------------------------------------------

--  DDL for Procedure CREAR_LINEAPEDIDO

-----------------------------------------------------

create or replace PROCEDURE CREAR_LINEAPEDIDO (

    cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,

    precio IN LINEAPEDIDOS.PRECIO%TYPE,

    producto IN LINEAPEDIDOS.CODIGO_PROD1%TYPE,

    pedido IN LINEAPEDIDOS.IDPEDIDO3%TYPE

  )

IS codigo LINEAPEDIDOS.NUMLINE%TYPE;

  BEGIN
```

```
    SELECT SEC_LINEAPEDIDOS.NEXTVAL INTO codigo
FROM dual;

    INSERT INTO LINEAPEDIDOS (NUMLINE,
CANTIDADPEDIDO, PRECIO, CODIGO_PROD1, IDPEDIDO3)

    VALUES(codigo, cant, precio, producto, pedido);

END CREAR_LINEAPEDIDO;

/

--------------------------------------------------

--  DDL for Procedure CREAR_PRODUCTO

--------------------------------------------------

create or replace PROCEDURE CREAR_PRODUCTO (

    nombre IN productos.NOMBRE%TYPE,

    descripcion IN productos.DESCRIPCION%TYPE,

    tipoproducto IN productos.TIPOPRODUCTO%TYPE,

    disponibilidad IN productos.DISPONIBILIDAD%TYPE,

    precio IN productos.PRECIO%TYPE,

    urldocu IN productos.URLDOCU%TYPE,

    urlimg IN productos.URLIMG%TYPE,

    stock IN productos.STOCK%TYPE,
```

```
    partnumber IN productos.PARTNUMBER%TYPE,
    modelo IN productos.MODELO%TYPE
  )
IS codigo productos.CODIGO%TYPE;
  BEGIN
    SELECT sec_producto.NEXTVAL INTO codigo FROM dual;
    INSERT INTO productos (CODIGO, NOMBRE,
DESCRIPCION, TIPOPRODUCTO, DISPONIBILIDAD, PRECIO,
URLDOCU, URLIMG, STOCK, PARTNUMBER, MODELO)
    VALUES(codigo, nombre, descripcion, tipoproducto,
disponibilidad, precio, urldocu, urlimg, stock, partnumber,
modelo);
END CREAR_PRODUCTO;
/
-----------------------------------------------
--  DDL for Procedure NUEVA_DIRECCION
-----------------------------------------------
create or replace PROCEDURE NUEVA_DIRECCION
( ciudad IN direcciones.CIUDAD%TYPE, numero IN
direcciones.NUMERO%TYPE, cp IN direcciones.CP%TYPE, calle
IN direcciones.CALLE%TYPE)
```

```
AS id_direccion direcciones.ID_DIRECCION%TYPE;


BEGIN

    SELECT sec_direccion.NEXTVAL INTO id_direccion FROM
dual;

    INSERT INTO direcciones (ID_DIRECCION, CIUDAD, CP,
CALLE, NUMERO)

    VALUES(id_direccion, ciudad, cp, calle, numero);

END NUEVA_DIRECCION;

/

-----------------------------------------------

-- Procedimientos de Actualizacion/modificacion

-----------------------------------------------



-----------------------------------------------

--  DDL for Procedure UPDATE_ALBARAN

-----------------------------------------------

create or replace PROCEDURE UPDATE_ALBARAN (

    id_albaranup IN ALBARANES.IDALBARAN%TYPE,
```

```
      id_pedidoup IN ALBARANES.IDPEDIDO2%TYPE
   )
IS
BEGIN
   UPDATE albaranes
   SET IDPEDIDO2 = id_pedidoup
   WHERE IDALBARAN = id_albaranup;
END UPDATE_ALBARAN;
/
-----------------------------------------------
--  DDL for Procedure UPDATE_ALMACEN
-----------------------------------------------
create or replace PROCEDURE UPDATE_ALMACEN (
   id_almacenup IN ALMACENES.IDALMACEN%TYPE,
   nombreup IN ALMACENES.NOMBRE%TYPE,
   id_direccionup IN ALMACENES.ID_DIRECCION3%TYPE
   )
IS
BEGIN
```

```
    UPDATE almacenes
    SET NOMBRE = nombreup,
      ID_DIRECCION3 = id_direccionup
    WHERE IDALMACEN = id_almacenup;
END UPDATE_ALMACEN;
/
--------------------------------------------------
--  DDL for Procedure UPDATE_CLIENTE
--------------------------------------------------
create or replace PROCEDURE UPDATE_CLIENTE (
  id_clienteUP IN CLIENTES.IDCLIENTE%TYPE,
  nombreUP IN CLIENTES.NOMBRE_CLI%TYPE,
  appUP IN CLIENTES.APP_CLI%TYPE,
  apmUP IN CLIENTES.APM_CLI%TYPE,
  emaUP IN CLIENTES.EMAIL%TYPE,
  dniUP IN CLIENTES.DNI%TYPE,
  telUP IN CLIENTES.TELEFONO%TYPE,
  id_direccion_CLIUP IN CLIENTES.ID_DIRECCION2%TYPE
  )
```

```
IS
BEGIN
    UPDATE clientes
    SET NOMBRE_CLI = nombreUP,
    APP_CLI = appUP,
    APM_CLI = apmUP,
    EMAIL = emaUP,
    DNI = dniUP,
    TELEFONO = telUP,
    ID_DIRECCION2 = id_direccion_CLIUP
    WHERE IDCLIENTE = id_clienteUP;
END UPDATE_CLIENTE;
/
-----------------------------------------------
--  DDL for Procedure UPDATE_OFERTA
-----------------------------------------------
create or replace PROCEDURE UPDATE_OFERTA
(      id_oferta IN OFERTAS.CODIGO%TYPE,
       fechaini IN OFERTAS.FECHA_INICIO%TYPE,
```

```
    fechafin IN OFERTAS.FECHA_FIN%TYPE,
    precioferta IN OFERTAS.PRECIOOFERTADO%TYPE,
    codigopro IN OFERTAS.CODIGO_PROD2 %TYPE,
    precioanterior IN OFERTAS.PRECIOANTES%TYPE)
IS
BEGIN
  UPDATE OFERTAS
  SET FECHA_INICIO = fechaini,
  FECHA_FIN = fechafin,
  PRECIOOFERTADO = precioferta,
  CODIGO_PROD2 = codigopro,
  PRECIOANTES = precioanterior
  WHERE CODIGO = id_oferta;
END UPDATE_OFERTA;
/
-----------------------------------------------------
--  DDL for Procedure UPDATE_DIRECCION
-----------------------------------------------------
create or replace PROCEDURE UPDATE_DIRECCION
```

```
(      id_direccionup IN DIRECCIONES.ID_DIRECCION%TYPE,
       ciudadup IN direcciones.CIUDAD%TYPE,
       cpup IN direcciones.CP%TYPE,
       calleup IN direcciones.CALLE%TYPE,
       numeroup IN direcciones.NUMERO%TYPE)
IS
BEGIN
    UPDATE direcciones
    SET ciudad = ciudadup,
    cp = cpup,
    calle = calleup,
    numero = numeroup
    WHERE ID_DIRECCION = id_direccionup;
END UPDATE_DIRECCION;
/
-----------------------------------------------------
--  DDL for Procedure UPDATE_ESTADOPEDIDO
-----------------------------------------------------
create or replace PROCEDURE UPDATE_ESTADOPEDIDO (
```

```
      pedido IN ESTADOPEDIDOS.IDPEDIDO1%TYPE,
      estado IN ESTADOPEDIDOS.ESTADOPEDIDO%TYPE
   )
IS
BEGIN
   UPDATE ESTADOPEDIDOS
   SET ESTADOPEDIDO = estado
   WHERE IDPEDIDO1 = pedido;
END UPDATE_ESTADOPEDIDO;
/
-----------------------------------------------------
--  DDL for Procedure UPDATE_FACTURA
-----------------------------------------------------
create or replace PROCEDURE UPDATE_FACTURA (
      id_factura IN FACTURAS.IDFACTURA%TYPE,
      precio IN FACTURAS.PRECIOTOTAL%TYPE,
      pago IN FACTURAS.TIPOFORMAPAGO%TYPE

   )
```

```
IS
BEGIN
   UPDATE FACTURAS
   SET PRECIOTOTAL = precio,
      TIPOFORMAPAGO = pago
   WHERE IDFACTURA = id_factura;
END UPDATE_FACTURA;
/
--------------------------------------------------
--  DDL for Procedure UPDATE_LINEAPEDIDO
--------------------------------------------------
create or replace PROCEDURE UPDATE_LINEAPEDIDO (
      num IN LINEAPEDIDOS.NUMLINE%TYPE,
      cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,
      precio IN LINEAPEDIDOS.PRECIO%TYPE,
      idProduc IN LINEAPEDIDOS.CODIGO_PROD1%TYPE
   )
IS
   BEGIN
```

```
   UPDATE LINEAPEDIDOS
   SET CANTIDADPEDIDO = cant,
   PRECIO = precio
   WHERE NUMLINE = num;
END UPDATE_LINEAPEDIDO;
/
--------------------------------------------------
--  DDL for Procedure UPDATE_PEDIDO
--------------------------------------------------
create or replace PROCEDURE UPDATE_PEDIDO (
      codigopedido IN PEDIDOS.IDPEDIDO%TYPE,
      fech IN PEDIDOS.FECHA%TYPE,
      subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,
      cliente IN PEDIDOS.IDCLIENTE1%TYPE,
      direccion IN PEDIDOS.ID_DIRECCION1%TYPE
   )
IS
   BEGIN
   UPDATE PEDIDOS
```

```
   SET PRECIOTOTAL = subtotal,
   ID_DIRECCION1 = direccion
   WHERE IDPEDIDO = codigopedido;
END UPDATE_PEDIDO;
/
--------------------------------------------------
--  DDL for Procedure UPDATE_PRODUCTO
--------------------------------------------------
create or replace PROCEDURE UPDATE_PRODUCTO (
            codigoup IN productos.CODIGO%TYPE,
      nombreup IN productos.NOMBRE%TYPE,
      descripcionup IN productos.DESCRIPCION%TYPE,
      tipoproductoup IN productos.TIPOPRODUCTO%TYPE,
      disponibilidadup IN productos.DISPONIBILIDAD%TYPE,
      precioup IN productos.PRECIO%TYPE,
      urldocuup IN productos.URLDOCU%TYPE,
      urlimgup IN productos.URLIMG%TYPE,
      stockup IN productos.STOCK%TYPE,
      partnumberup IN productos.PARTNUMBER%TYPE,
```

```
        modeloup IN productos.MODELO%TYPE
    )
IS
    BEGIN
     UPDATE productos
     SET nombre = nombreup,
     descripcion = descripcionup,
     tipoproducto = tipoproductoup,
     disponibilidad = disponibilidadup,
     precio = precioup,
     urldocu = urldocuup,
     urlimg = urlimgup,
     stock = stockup,
     partnumber = partnumberup,
     modelo = modeloup
     WHERE codigo = codigoup;
END UPDATE_PRODUCTO;
/
 --Procedimiento para reiniciar una secuencia
```

```
CREATE OR REPLACE PROCEDURE RESET_SEC( p_seq_name
in varchar2 ) is

        l_val number;

    BEGIN

        execute immediate

         'select ' || p_seq_name || '.nextval from dual' INTO l_val;

         execute immediate

        'alter sequence ' || p_seq_name || ' increment by -' || l_val
|| ' minvalue 0';

        execute immediate

        'select ' || p_seq_name || '.nextval from dual' INTO l_val;

        execute immediate

       'alter sequence ' || p_seq_name || ' increment by 1 minvalue
0';

   END;
```

Secuencias.

--Secuencias

```
--Secuencia IDCLIENTE

CREATE   SEQUENCE sec_cliente

 INCREMENT BY 1 START WITH 1;


 --Secuencia PRODUCTO

CREATE   SEQUENCE sec_producto

 INCREMENT BY 1 START WITH 1;


 --Secuencia IDPEDIDO

CREATE  SEQUENCE sec_pedido

 INCREMENT BY 1 START WITH 1;


--Secuencia IDALBARAN

CREATE  SEQUENCE sec_albaran

 INCREMENT BY 1 START WITH 1;


--Secuencia IDFACTURA

CREATE SEQUENCE sec_factura

 INCREMENT BY 1 START WITH 1;
```

--Secuencia LINEAPEDIDOS

CREATE SEQUENCE sec_lineaPedidos

INCREMENT BY 1 START WITH 1;

--Secuencia IDOFERTA

CREATE SEQUENCE sec_oferta

INCREMENT BY 1 START WITH 1;

--Secuencia ID_DIRECCION

CREATE SEQUENCE sec_direccion

INCREMENT BY 1 START WITH 1;

--Secuencia ID_ALMACEN

CREATE SEQUENCE sec_almacen

INCREMENT BY 1 START WITH 1;

Triggers.

--Triggers Reglas de negocio

```
--Trigger RN01

 CREATE OR REPLACE TRIGGER datosCliente

   BEFORE DELETE ON CLIENTES

   FOR EACH ROW

   DECLARE

      estado VARCHAR(20);

      BEGIN

         SELECT  ESTADOPEDIDO INTO estado FROM
ESTADOPEDIDOS WHERE ESTADOPEDIDO != 'Enviado';

         IF(estado != 'Enviado')

            THEN Raise_application_error(-20600,'No se puede
eliminar un cliente, tiene pedidos pendientes.');

   END IF;

   END;

/

--Trigger RN02

 create or replace TRIGGER horaPedido

   BEFORE INSERT ON PEDIDOS

   FOR EACH ROW
```

```
DECLARE
        prueba VARCHAR2(10);
        horaMax VARCHAR2(50):= '17:00';
 BEGIN
        select to_char(sysdate, 'HH24:MI') into prueba  from dual;
        if prueba<=horaMax then
                UPDATE ESTADOPEDIDOS
      SET ESTADOPEDIDO = 'En proceso';
        ELSE
                UPDATE ESTADOPEDIDOS
      SET ESTADOPEDIDO = 'En espera';
 END IF;
END;
/
 --Trigger RN03
create or replace TRIGGER COMPRUEBAPAGO
 BEFORE UPDATE ON ESTADOPEDIDOS
 FOR EACH ROW
BEGIN
```

```
   IF (:NEW.ESTADOPEDIDO = 'En proceso')

   THEN IF (:OLD.ESTADOPEDIDO = 'Pendiente de pago')

   THEN Raise_application_error(-20600,'No se puede procesar un
pedido pendiente de pago');

   END IF;

   END IF;

 END;

/

 --Trigger RN04

create or replace TRIGGER envioPedido

   BEFORE UPDATE ON ESTADOPEDIDOS

      FOR EACH ROW

   BEGIN

   IF (:NEW.ESTADOPEDIDO = 'Enviado')

   THEN IF (:OLD.ESTADOPEDIDO = 'Pendiente de Pago')

   THEN Raise_application_error(-20600,'No se puede enviar un
pedido pendiente de pago');

   END IF;

   END IF;
```

```
 END;

/

 --Triggers asociados a las secuencias

 CREATE OR REPLACE TRIGGER crea_ID_almacen

  BEFORE INSERT ON ALMACENES

  FOR EACH ROW

  BEGIN

  SELECT sec_albaran.NEXTVAL INTO :NEW.IDALMACEN
FROM DUAL;

  END;

  /


 CREATE OR REPLACE TRIGGER crea_ID_cliente

  BEFORE INSERT ON CLIENTES

  FOR EACH ROW

  BEGIN

  SELECT sec_cliente.NEXTVAL INTO :NEW.IDCLIENTE FROM
DUAL;

  END;
```

```
/

CREATE OR REPLACE TRIGGER crea_ID_producto

 BEFORE INSERT ON PRODUCTOS

 FOR EACH ROW

 BEGIN

 SELECT sec_producto.NEXTVAL INTO :NEW.CODIGO FROM
DUAL;

 END;

/

CREATE OR REPLACE TRIGGER crea_ID_pedido

 BEFORE INSERT ON PEDIDOS

 FOR EACH ROW

 BEGIN

 SELECT sec_pedido.NEXTVAL INTO :NEW.IDPEDIDO FROM
DUAL;

 END;

/

CREATE OR REPLACE TRIGGER crea_ID_albaran

 BEFORE INSERT ON ALBARANES
```

```
  FOR EACH ROW

  BEGIN

  SELECT sec_albaran.NEXTVAL INTO :NEW.IDALBARAN
FROM DUAL;

  END;

/

  CREATE OR REPLACE TRIGGER crea_LINEAPEDIDOS

  BEFORE INSERT ON LINEAPEDIDOS

  FOR EACH ROW

  BEGIN

  SELECT sec_LINEAPEDIDOS.NEXTVAL INTO :NEW.numline
FROM DUAL;

  END;

/

CREATE OR REPLACE TRIGGER crea_ID_factura

  BEFORE INSERT ON FACTURAS

  FOR EACH ROW

  BEGIN

  SELECT sec_factura.NEXTVAL INTO :NEW.IDFACTURA FROM
DUAL;
```

```
  END;
/
CREATE OR REPLACE TRIGGER crea_ID_oferta
  BEFORE INSERT ON OFERTAS
  FOR EACH ROW
  BEGIN
  SELECT sec_oferta.NEXTVAL INTO :NEW.CODIGO FROM
DUAL;
  END;
/
CREATE OR REPLACE TRIGGER crea_ID_direccion
  BEFORE INSERT ON DIRECCIONES
  FOR EACH ROW
  BEGIN
  SELECT sec_direccion.NEXTVAL INTO :NEW.ID_DIRECCION
FROM DUAL;
  END;
/
```

Paquetes de Prueba

```
-- Pruebas de ALMACENES

----------------------------------------------

create or replace PACKAGE ALMACENES_PRUEBA IS

/*Creamos el procedimiento inicializar la cabecera*/


 Procedure INICIALIZAR;

 /*Creamos el procedimiento crear la cabecera*/

   /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE INSERTAR (

   nombre_prueba_in VARCHAR2,

   nombre IN ALMACENES.NOMBRE%TYPE,

   id_direccion IN ALMACENES.ID_DIRECCION3%TYPE,

   salida_esperada_in BOOLEAN

   );

 /*Creamos el procedimiento borrar la cabecera*/

   /* Formato procedimientos:
```

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

PROCEDURE ELIMINAR (

nombre_prueba_el VARCHAR2,

id_almacen IN ALMACENES.IDALMACEN%TYPE,

salida_esperada_el BOOLEAN

);

/*Creamos el procedimiento actualizar la cabecera*/

/* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

PROCEDURE ACTUALIZAR (

nombre_prueba_up VARCHAR2,

id_almacenup IN ALMACENES.IDALMACEN%TYPE,

nombreup IN ALMACENES.NOMBRE%TYPE,

id_direccionup IN ALMACENES.ID_DIRECCION3%TYPE,

salida_esperada_up BOOLEAN

);

```
END ALMACENES_PRUEBA;

/

create or replace PACKAGE BODY ALMACENES_PRUEBA AS

 procedure INICIALIZAR is

 begin

 /*Se borran todas las tablas*/


 DELETE FROM ALMACENES;

 end INICIALIZAR;


 /*Copiar y pegar el procedimiento crear*/


 PROCEDURE INSERTAR (

   nombre_prueba_in VARCHAR2,

   nombre IN ALMACENES.NOMBRE%TYPE,

   id_direccion IN ALMACENES.ID_DIRECCION3%TYPE,

   salida_esperada_in BOOLEAN

   )
```

```
AS salidain BOOLEAN := TRUE;
   almacen ALMACENES.NOMBRE%TYPE;
   id_almacen ALMACENES.IDALMACEN%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/
      CREAR_ALMACEN(nombre,id_direccion);
      id_almacen := SEC_ALMACEN.CURRVAL;
      /*Comprobar que se ha realizado con exito */
      SELECT ALMACENES.NOMBRE INTO almacen FROM
ALMACENES WHERE IDALMACEN = id_almacen;
      IF(almacen<>nombre ) THEN
      salidain := FALSE;
      END IF;
      COMMIT WORK;
   /*Mostrar resultados de la prueba*/
      DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
      || ASSERT_EQUALS(salidain,salida_Esperada_in));

      EXCEPTION
```

```
      WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
       || ASSERT_EQUALS(false,salida_Esperada_in));
       ROLLBACK;
      END INSERTAR;

 /*Copiar y pegar el procedimiento borrar*/
   PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   id_almacen IN ALMACENES.IDALMACEN%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numeroAlmacenes NUMBER;
BEGIN
   BORRAR_ALMACEN(id_almacen);
   SELECT COUNT(*) INTO numeroAlmacenes FROM
ALMACENES
   WHERE IDALMACEN= id_almacen;
```

```
IF(numeroAlmacenes <> 0) THEN
salidael:= FALSE;
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(salidael,salida_Esperada_el));

  EXCEPTION
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(false,salida_Esperada_el));
   ROLLBACK;
END ELIMINAR;
/*Copiar y pegar el procedimiento actualizar*/

PROCEDURE ACTUALIZAR (
 nombre_prueba_up VARCHAR2,
 id_almacenup IN ALMACENES.IDALMACEN%TYPE,
 nombreup IN ALMACENES.NOMBRE%TYPE,
```

```
   id_direccionup IN ALMACENES.ID_DIRECCION3%TYPE,
   salida_esperada_up BOOLEAN
   )
AS salidaup BOOLEAN := TRUE;
   almacen ALMACENES.NOMBRE%TYPE;
   id_almacen ALMACENES.IDALMACEN%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/

UPDATE_ALMACEN(id_almacenup,nombreup,id_direccionup);
      id_almacen := SEC_ALMACEN.CURRVAL;
      /*Comprobar que se ha realizado con exito */
      SELECT ALMACENES.NOMBRE INTO almacen FROM
ALMACENES WHERE IDALMACEN = id_almacen;
      IF(almacen <> nombreup ) THEN
      salidaup := FALSE;
      END IF;
      COMMIT WORK;
```

```
   /*Mostrar resultados de la prueba*/
      DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
      || ASSERT_EQUALS(salidaup,salida_Esperada_up));

      EXCEPTION
      WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
       || ASSERT_EQUALS(false,salida_Esperada_up));
       ROLLBACK;
      END ACTUALIZAR;

   end ALMACENES_PRUEBA;
/
-------------------------------------------------
-- Pruebas de LINEAPEDIDOS
-------------------------------------------------
   create or replace PACKAGE LINEAPEDIDOS_PRUEBA IS
/*Creamos el procedimiento inicializar la cabecera*/
```

```
Procedure INICIALIZAR;

/*Creamos el procedimiento crear la cabecera*/

  /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

PROCEDURE INSERTAR (

  nombre_prueba_in VARCHAR2,

  cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,

  precio IN LINEAPEDIDOS.PRECIO%TYPE,

  producto IN LINEAPEDIDOS.CODIGO_PROD1%TYPE,

  pedido IN LINEAPEDIDOS.IDPEDIDO3%TYPE,

  salida_esperada_in BOOLEAN

  );

/*Creamos el procedimiento borrar la cabecera*/

  /* Formato procedimientos:


nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

PROCEDURE ELIMINAR (
```

```
   nombre_prueba_el VARCHAR2,
   num IN LINEAPEDIDOS.NUMLINE%TYPE,
   salida_esperada_el BOOLEAN
   );
/*Creamos el procedimiento actualizar la cabecera*/
/* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE ACTUALIZAR (
   nombre_prueba_up VARCHAR2,
   num IN LINEAPEDIDOS.NUMLINE%TYPE,
   cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,
   precio IN LINEAPEDIDOS.PRECIO%TYPE,
   idProduc IN LINEAPEDIDOS.CODIGO_PROD1%TYPE,
   salida_esperada_up BOOLEAN
   );

END LINEAPEDIDOS_PRUEBA;
```

```
/

create or replace PACKAGE BODY LINEAPEDIDOS_PRUEBA AS

 procedure INICIALIZAR is

 begin

 /*Se borran todas las tablas*/


 DELETE FROM LINEAPEDIDOS;

 end INICIALIZAR;


 /*Copiar y pegar el procedimiento crear*/


 PROCEDURE INSERTAR (

   nombre_prueba_in VARCHAR2,

   cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,

   precio IN LINEAPEDIDOS.PRECIO%TYPE,

   producto IN LINEAPEDIDOS.CODIGO_PROD1%TYPE,

   pedido IN LINEAPEDIDOS.IDPEDIDO3%TYPE,

   salida_esperada_in BOOLEAN

   )
```

```
AS salidain BOOLEAN := TRUE;
   linea LINEAPEDIDOS.NUMLINE%TYPE;
   id_LINEAPEDIDO LINEAPEDIDOS.NUMLINE%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/
      CREAR_LINEAPEDIDO(cant,precio,producto,pedido);
      id_LINEAPEDIDO := SEC_LINEAPEDIDOS.CURRVAL;
      /*Comprobar que se ha realizado con exito */
      SELECT IDPEDIDO3 INTO linea FROM LINEAPEDIDOS
WHERE NUMLINE = id_LINEAPEDIDO;
      IF(linea<>pedido ) THEN
      salidain := FALSE;
      END IF;
      COMMIT WORK;
   /*Mostrar resultados de la prueba*/
      DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
      || ASSERT_EQUALS(salidain,salida_Esperada_in));

      EXCEPTION
```

```
      WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
       || ASSERT_EQUALS(false,salida_Esperada_in));
       ROLLBACK;
      END INSERTAR;

 /*Copiar y pegar el procedimiento borrar*/
   PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   num IN LINEAPEDIDOS.NUMLINE%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numerolineas NUMBER;
BEGIN
   ELIMINA_LINEAPEDIDO(num);
   SELECT COUNT(*) INTO numerolineas FROM
LINEAPEDIDOS
   WHERE NUMLINE= num;
```

```
IF(numerolineas <> 0) THEN
salidael:= FALSE;
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(salidael,salida_Esperada_el));

  EXCEPTION
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(false,salida_Esperada_el));
   ROLLBACK;
END ELIMINAR;
/*Copiar y pegar el procedimiento actualizar*/

PROCEDURE ACTUALIZAR (
nombre_prueba_up VARCHAR2,
num IN LINEAPEDIDOS.NUMLINE%TYPE,
cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,
```

```
   precio IN LINEAPEDIDOS.PRECIO%TYPE,
   idProduc IN LINEAPEDIDOS.CODIGO_PROD1%TYPE,
   salida_esperada_up BOOLEAN
   )
AS salidaup BOOLEAN := TRUE;
   linea LINEAPEDIDOS.CODIGO_PROD1%TYPE;
   id_linea LINEAPEDIDOS.NUMLINE%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/
      UPDATE_LINEAPEDIDO(num,cant,precio,idProduc);
      id_linea := SEC_LINEAPEDIDOS.CURRVAL;
      /*Comprobar que se ha realizado con exito */
      SELECT CODIGO_PROD1 INTO linea FROM
LINEAPEDIDOS WHERE NUMLINE = num;
      IF(idProduc <> linea ) THEN
      salidaup := FALSE;
      END IF;
      COMMIT WORK;
```

```
/*Mostrar resultados de la prueba*/
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
    || ASSERT_EQUALS(salidaup,salida_Esperada_up));

    EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
     || ASSERT_EQUALS(false,salida_Esperada_up));
     ROLLBACK;
    END ACTUALIZAR;

  end LINEAPEDIDOS_PRUEBA;
  /
 ------------------------------------------------
-- Pruebas de PEDIDOS
-------------------------------------------------
create or replace PACKAGE PEDIDOS_PRUEBA IS
/*Creamos el procedimiento inicializar la cabecera*/
```

Procedure INICIALIZAR;

/*Creamos el procedimiento crear la cabecera*/

/* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

PROCEDURE INSERTAR (

nombre_prueba_in VARCHAR2,

subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,

cliente IN PEDIDOS.IDCLIENTE1%TYPE,

direccion IN PEDIDOS.ID_DIRECCION1%TYPE,

salida_esperada_in BOOLEAN

);

/*Creamos el procedimiento borrar la cabecera*/

/* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

PROCEDURE ELIMINAR (

nombre_prueba_el VARCHAR2,

```
   codigoborrar IN PEDIDOS.IDPEDIDO%TYPE,
   salida_esperada_el BOOLEAN
   );
/*Creamos el procedimiento actualizar la cabecera*/
/* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE ACTUALIZAR (
   nombre_prueba_up VARCHAR2,
   codigopedido IN PEDIDOS.IDPEDIDO%TYPE,
   fech IN PEDIDOS.FECHA%TYPE,
   subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,
   cliente IN PEDIDOS.IDCLIENTE1%TYPE,
   direccion IN PEDIDOS.ID_DIRECCION1%TYPE,
   salida_esperada_up BOOLEAN
   );

END PEDIDOS_PRUEBA;
```

```
/

create or replace PACKAGE BODY PEDIDOS_PRUEBA AS

 procedure INICIALIZAR is

 begin

 /*Se borran todas las tablas*/


 DELETE FROM PEDIDOS;

 end INICIALIZAR;


 /*Copiar y pegar el procedimiento crear*/


 PROCEDURE INSERTAR (

   nombre_prueba_in VARCHAR2,

   subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,

   cliente IN PEDIDOS.IDCLIENTE1%TYPE,

   direccion IN PEDIDOS.ID_DIRECCION1%TYPE,

   salida_esperada_in BOOLEAN

   )

AS salidain BOOLEAN := TRUE;
```

```
    pedido PEDIDOS.IDPEDIDO%TYPE;
    id_PEDIDO PEDIDOS.IDPEDIDO%TYPE;
BEGIN
    /*Llamada al procedimiento a probar*/
    CREAR_PEDIDO(subtotal,cliente,direccion);
    id_PEDIDO := SEC_PEDIDO.CURRVAL;
    /*Comprobar que se ha realizado con exito */
    SELECT IDPEDIDO INTO pedido FROM PEDIDOS WHERE
IDPEDIDO = id_PEDIDO;
    IF(id_PEDIDO<>pedido ) THEN
    salidain := FALSE;
    END IF;
    COMMIT WORK;
  /*Mostrar resultados de la prueba*/
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
    || ASSERT_EQUALS(salidain,salida_Esperada_in));

    EXCEPTION
    WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
        || ASSERT_EQUALS(false,salida_Esperada_in));
        ROLLBACK;
       END INSERTAR;


 /*Copiar y pegar el procedimiento borrar*/
   PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   codigoborrar IN PEDIDOS.IDPEDIDO%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numeropedidos NUMBER;
BEGIN
   ELIMINA_PEDIDO(codigoborrar);
   SELECT COUNT(*) INTO numeropedidos FROM PEDIDOS
   WHERE IDPEDIDO = codigoborrar;
   IF(numeropedidos <> 0) THEN
   salidael:= FALSE;
```

```
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(salidael,salida_Esperada_el));

  EXCEPTION
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(false,salida_Esperada_el));
   ROLLBACK;
END ELIMINAR;
/*Copiar y pegar el procedimiento actualizar*/

PROCEDURE ACTUALIZAR (
 nombre_prueba_up VARCHAR2,
 codigopedido IN PEDIDOS.IDPEDIDO%TYPE,
 fech IN PEDIDOS.FECHA%TYPE,
 subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,
 cliente IN PEDIDOS.IDCLIENTE1%TYPE,
```

```
   direccion IN PEDIDOS.ID_DIRECCION1%TYPE,
   salida_esperada_up BOOLEAN
   )
AS salidaup BOOLEAN := TRUE;
   pedido PEDIDOS.IDPEDIDO%TYPE;
   id_pedido PEDIDOS.IDPEDIDO%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/

UPDATE_PEDIDO(codigopedido,fech,subtotal,cliente,direccion);
      id_pedido := SEC_PEDIDO.CURRVAL;
      /*Comprobar que se ha realizado con exito */
      SELECT IDPEDIDO INTO pedido FROM PEDIDOS WHERE
IDPEDIDO = id_pedido;
      IF(id_pedido <> pedido ) THEN
      salidaup := FALSE;
      END IF;
      COMMIT WORK;
```

```
   /*Mostrar resultados de la prueba*/
      DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
      || ASSERT_EQUALS(salidaup,salida_Esperada_up));

      EXCEPTION
      WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
       || ASSERT_EQUALS(false,salida_Esperada_up));
       ROLLBACK;
      END ACTUALIZAR;

   end PEDIDOS_PRUEBA;
/
-------------------------------------------------
-- Pruebas de PRODUCTOS
-------------------------------------------------
   create or replace PACKAGE PRODUCTOS_PRUEBA IS
/*Creamos el procedimiento inicializar la cabecera*/
```

```
Procedure INICIALIZAR;

/*Creamos el procedimiento crear la cabecera*/

  /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

PROCEDURE INSERTAR (

  nombre_prueba_in VARCHAR2,

  nombre IN productos.NOMBRE%TYPE,

  descripcion IN productos.DESCRIPCION%TYPE,

  tipoproducto IN productos.TIPOPRODUCTO%TYPE,

  disponibilidad IN productos.DISPONIBILIDAD%TYPE,

  precio IN productos.PRECIO%TYPE,

  urldocu IN productos.URLDOCU%TYPE,

  urlimg IN productos.URLIMG%TYPE,

  stock IN productos.STOCK%TYPE,

  partnumber IN productos.PARTNUMBER%TYPE,

  modelo IN productos.MODELO%TYPE,

  salida_esperada_in BOOLEAN
```

```
    );
/*Creamos el procedimiento borrar la cabecera*/
   /* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   codigoborrar IN productos.CODIGO%TYPE,
   salida_esperada_el BOOLEAN
   );
/*Creamos el procedimiento actualizar la cabecera*/
/* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE ACTUALIZAR (
   nombre_prueba_up VARCHAR2,
   codigoup IN productos.CODIGO%TYPE,
   nombreup IN productos.NOMBRE%TYPE,
```

```
    descripcionup IN productos.DESCRIPCION%TYPE,
    tipoproductoup IN productos.TIPOPRODUCTO%TYPE,
    disponibilidadup IN productos.DISPONIBILIDAD%TYPE,
    precioup IN productos.PRECIO%TYPE,
    urldocuup IN productos.URLDOCU%TYPE,
    urlimgup IN productos.URLIMG%TYPE,
    stockup IN productos.STOCK%TYPE,
    partnumberup IN productos.PARTNUMBER%TYPE,
    modeloup IN productos.MODELO%TYPE,
    salida_esperada_up BOOLEAN
    );

END PRODUCTOS_PRUEBA;
/
create or replace PACKAGE BODY PRODUCTOS_PRUEBA AS
 procedure INICIALIZAR is
 begin
 /*Se borran todas las tablas*/
```

```
DELETE FROM PRODUCTOS;

end INICIALIZAR;


/*Copiar y pegar el procedimiento crear*/


PROCEDURE INSERTAR (
  nombre_prueba_in VARCHAR2,
  nombre IN productos.NOMBRE%TYPE,
  descripcion IN productos.DESCRIPCION%TYPE,
  tipoproducto IN productos.TIPOPRODUCTO%TYPE,
  disponibilidad IN productos.DISPONIBILIDAD%TYPE,
  precio IN productos.PRECIO%TYPE,
  urldocu IN productos.URLDOCU%TYPE,
  urlimg IN productos.URLIMG%TYPE,
  stock IN productos.STOCK%TYPE,
  partnumber IN productos.PARTNUMBER%TYPE,
  modelo IN productos.MODELO%TYPE,
  salida_esperada_in BOOLEAN
  )
```

```
AS salidain BOOLEAN := TRUE;
   producto PRODUCTOS.CODIGO%TYPE;
   id_Producto PRODUCTOS.CODIGO%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/

CREAR_PRODUCTO(nombre,descripcion,tipoproducto,disponibil
idad,precio,urldocu,urlimg,stock,partnumber,modelo);
      id_Producto := SEC_Producto.CURRVAL;
      /*Comprobar que se ha realizado con exito */
      SELECT CODIGO INTO producto FROM PRODUCTOS
WHERE CODIGO = id_Producto;
      IF(id_Producto<>producto ) THEN
      salidain := FALSE;
      END IF;
      COMMIT WORK;
   /*Mostrar resultados de la prueba*/
      DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
      || ASSERT_EQUALS(salidain,salida_Esperada_in));
```

```
      EXCEPTION
      WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
       || ASSERT_EQUALS(false,salida_Esperada_in));
       ROLLBACK;
      END INSERTAR;


 /*Copiar y pegar el procedimiento borrar*/
   PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   codigoborrar IN productos.CODIGO%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numeroproductos NUMBER;
BEGIN
   ELIMINA_PRODUCTO(codigoborrar);
   SELECT COUNT(*) INTO numeroproductos FROM
PRODUCTOS
```

```
WHERE CODIGO = codigoborrar;
IF(numeroproductos <> 0) THEN
salidael:= FALSE;
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(salidael,salida_Esperada_el));

  EXCEPTION
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(false,salida_Esperada_el));
   ROLLBACK;
END ELIMINAR;
/*Copiar y pegar el procedimiento actualizar*/

PROCEDURE ACTUALIZAR (
 nombre_prueba_up VARCHAR2,
 codigoup IN productos.CODIGO%TYPE,
```

```
    nombreup IN productos.NOMBRE%TYPE,
    descripcionup IN productos.DESCRIPCION%TYPE,
    tipoproductoup IN productos.TIPOPRODUCTO%TYPE,
    disponibilidadup IN productos.DISPONIBILIDAD%TYPE,
    precioup IN productos.PRECIO%TYPE,
    urldocuup IN productos.URLDOCU%TYPE,
    urlimgup IN productos.URLIMG%TYPE,
    stockup IN productos.STOCK%TYPE,
    partnumberup IN productos.PARTNUMBER%TYPE,
    modeloup IN productos.MODELO%TYPE,
    salida_esperada_up BOOLEAN
    )
AS salidaup BOOLEAN := TRUE;
    producto PRODUCTOS.CODIGO %TYPE;
    id_producto PRODUCTOS.CODIGO%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/
```

```
UPDATE_PRODUCTO(codigoup,nombreup,descripcionup,tipopro
ductoup,

    disponibilidadup,precioup,
urldocuup,urlimgup,stockup,partnumberup,modeloup);

    id_producto := SEC_PRODUCTO.CURRVAL;

    /*Comprobar que se ha realizado con exito */

    SELECT CODIGO INTO producto FROM PRODUCTOS
WHERE CODIGO = id_producto;

    IF(id_producto <> producto ) THEN

    salidaup := FALSE;

    END IF;

    COMMIT WORK;



  /*Mostrar resultados de la prueba*/

    DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '

    || ASSERT_EQUALS(salidaup,salida_Esperada_up));


    EXCEPTION
```

```
      WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
       || ASSERT_EQUALS(false,salida_Esperada_up));
       ROLLBACK;
      END ACTUALIZAR;

   end PRODUCTOS_PRUEBA;
   /
-------------------------------------------------
-- Pruebas de FACTURAS
-------------------------------------------------
create or replace PACKAGE FACTURAS_PRUEBA IS
/*Creamos el procedimiento inicializar la cabecera*/

 Procedure INICIALIZAR;
 /*Creamos el procedimiento crear la cabecera*/
   /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
```

```
PROCEDURE INSERTAR (
  nombre_prueba_in VARCHAR2,
  albaran IN FACTURAS.IDALBARAN1%TYPE,
  pago IN FACTURAS.TIPOFORMAPAGO%TYPE,
  salida_esperada_in BOOLEAN
  );
/*Creamos el procedimiento borrar la cabecera*/
  /* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
PROCEDURE ELIMINAR (
  nombre_prueba_el VARCHAR2,
  idborrar IN FACTURAS.IDFACTURA%TYPE,
  salida_esperada_el BOOLEAN
  );
/*Creamos el procedimiento actualizar la cabecera*/
/* Formato procedimientos:
```

```
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE ACTUALIZAR (

   nombre_prueba_up VARCHAR2,

   id_facturaup IN FACTURAS.IDFACTURA%TYPE,

   precio IN FACTURAS.PRECIOTOTAL%TYPE,

   pago IN FACTURAS.TIPOFORMAPAGO%TYPE,

   salida_esperada_up BOOLEAN

   );


END FACTURAS_PRUEBA;

/

create or replace PACKAGE BODY FACTURAS_PRUEBA AS

 procedure INICIALIZAR is

 begin

 /*Se borran todas las tablas*/


 DELETE FROM FACTURAS;
```

```
end INICIALIZAR;


/*Copiar y pegar el procedimiento crear*/


PROCEDURE INSERTAR (
  nombre_prueba_in VARCHAR2,
  albaran IN FACTURAS.IDALBARAN1%TYPE,
  pago IN FACTURAS.TIPOFORMAPAGO%TYPE,
  salida_esperada_in BOOLEAN
  )
AS salidain BOOLEAN := TRUE;
  factura FACTURAS.IDFACTURA%TYPE;
  id_factura FACTURAS.IDFACTURA%TYPE;
BEGIN
    /*Llamada al procedimiento a probar*/
    CREAR_FACTURA(albaran,pago);
    id_factura := SEC_FACTURA.CURRVAL;
    /*Comprobar que se ha realizado con exito */
```

```
      SELECT IDFACTURA INTO factura FROM FACTURAS
WHERE IDFACTURA = id_factura;

      IF(id_factura<>factura ) THEN

      salidain := FALSE;

      END IF;

      COMMIT WORK;

   /*Mostrar resultados de la prueba*/

      DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '

      || ASSERT_EQUALS(salidain,salida_Esperada_in));


      EXCEPTION

      WHEN OTHERS THEN

       DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '

       || ASSERT_EQUALS(false,salida_Esperada_in));

       ROLLBACK;

      END INSERTAR;


  /*Copiar y pegar el procedimiento borrar*/

    PROCEDURE ELIMINAR (
```

```
   nombre_prueba_el VARCHAR2,
   idborrar IN FACTURAS.IDFACTURA%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numerofactura NUMBER;
BEGIN
   BORRAR_FACTURA(idborrar);
   SELECT COUNT(*) INTO numerofactura FROM FACTURAS
   WHERE IDFACTURA = idborrar;
   IF(numerofactura <> 0) THEN
   salidael:= FALSE;
   END IF;
   COMMIT WORK;
   DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
      || ASSERT_EQUALS(salidael,salida_Esperada_el));

      EXCEPTION
      WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
        || ASSERT_EQUALS(false,salida_Esperada_el));
        ROLLBACK;
    END ELIMINAR;
/*Copiar y pegar el procedimiento actualizar*/

 PROCEDURE ACTUALIZAR (
    nombre_prueba_up VARCHAR2,
    id_facturaup IN FACTURAS.IDFACTURA%TYPE,
    precio IN FACTURAS.PRECIOTOTAL%TYPE,
    pago IN FACTURAS.TIPOFORMAPAGO%TYPE,
    salida_esperada_up BOOLEAN
    )
AS salidaup BOOLEAN := TRUE;
    factura FACTURAS.IDFACTURA%TYPE;
    id_factura FACTURAS.IDFACTURA%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/
      UPDATE_FACTURA(id_facturaup,precio,pago);
```

```
id_factura := SEC_FACTURA.CURRVAL;

/*Comprobar que se ha realizado con exito */

SELECT IDFACTURA INTO factura FROM FACTURAS
WHERE IDFACTURA = id_facturaup;

IF(id_factura <> factura ) THEN

salidaup := FALSE;

END IF;

COMMIT WORK;



/*Mostrar resultados de la prueba*/

DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '

|| ASSERT_EQUALS(salidaup,salida_Esperada_up));


EXCEPTION

WHEN OTHERS THEN

 DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '

 || ASSERT_EQUALS(false,salida_Esperada_up));

 ROLLBACK;
```

```
      END ACTUALIZAR;


   end FACTURAS_PRUEBA;
/
-------------------------------------------------
-- Pruebas de OFERTAS
-------------------------------------------------
create or replace PACKAGE OFERTAS_PRUEBA IS
/*Creamos el procedimiento inicializar la cabecera*/

 Procedure INICIALIZAR;
 /*Creamos el procedimiento crear la cabecera*/
   /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE INSERTAR (
   nombre_prueba_in VARCHAR2,
   fechaini IN OFERTAS.FECHA_INICIO%TYPE,
   fechafin IN OFERTAS.FECHA_FIN%TYPE,
```

```
   precioDes IN OFERTAS.PRECIOOFERTADO%TYPE,
   codigoPro IN OFERTAS.CODIGO%TYPE,
   salida_esperada_in BOOLEAN
   );
/*Creamos el procedimiento borrar la cabecera*/
   /* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   id_ofertaborrar IN OFERTAS.CODIGO%TYPE,
   salida_esperada_el BOOLEAN
   );
/*Creamos el procedimiento actualizar la cabecera*/
/* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE ACTUALIZAR (
```

```
  nombre_prueba_up VARCHAR2,
  id_oferta IN OFERTAS.CODIGO%TYPE,
  fechaini IN OFERTAS.FECHA_INICIO%TYPE,
  fechafin IN OFERTAS.FECHA_FIN%TYPE,
  precioferta IN OFERTAS.PRECIOOFERTADO%TYPE,
  codigopro IN OFERTAS.CODIGO_PROD2 %TYPE,
  precioanterior IN OFERTAS.PRECIOANTES%TYPE,
  salida_esperada_up BOOLEAN
  );

END OFERTAS_PRUEBA;
/
create or replace PACKAGE BODY OFERTAS_PRUEBA AS
 procedure INICIALIZAR is
 begin
 /*Se borran todas las tablas*/

 DELETE FROM OFERTAS;
 end INICIALIZAR;
```

```
/*Copiar y pegar el procedimiento crear*/


PROCEDURE INSERTAR (
  nombre_prueba_in VARCHAR2,
  fechaini IN OFERTAS.FECHA_INICIO%TYPE,
  fechafin IN OFERTAS.FECHA_FIN%TYPE,
  precioDes IN OFERTAS.PRECIOOFERTADO%TYPE,
  codigoPro IN OFERTAS.CODIGO%TYPE,
  salida_esperada_in BOOLEAN
  )
AS salidain BOOLEAN := TRUE;
  oferta OFERTAS.CODIGO%TYPE;
  id_oferta OFERTAS.CODIGO%TYPE;
BEGIN
    /*Llamada al procedimiento a probar*/
    CREAR_OFERTA(fechaini,fechafin,precioDes,codigoPro);
    id_oferta := SEC_OFERTA.CURRVAL;
    /*Comprobar que se ha realizado con exito */
```

```
      SELECT CODIGO INTO oferta FROM OFERTAS WHERE
CODIGO = id_oferta;

      IF(id_oferta<>oferta ) THEN

      salidain := FALSE;

      END IF;

      COMMIT WORK;

   /*Mostrar resultados de la prueba*/

      DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '

      || ASSERT_EQUALS(salidain,salida_Esperada_in));


      EXCEPTION

      WHEN OTHERS THEN

       DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '

       || ASSERT_EQUALS(false,salida_Esperada_in));

       ROLLBACK;

      END INSERTAR;


  /*Copiar y pegar el procedimiento borrar*/

    PROCEDURE ELIMINAR (
```

```
   nombre_prueba_el VARCHAR2,
   id_ofertaborrar IN OFERTAS.CODIGO%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numeroferta NUMBER;
BEGIN
   ELIMINA_OFERTA(id_ofertaborrar);
   SELECT COUNT(*) INTO numeroferta FROM OFERTAS
   WHERE CODIGO = id_ofertaborrar;
   IF(numeroferta <> 0) THEN
   salidael:= FALSE;
   END IF;
   COMMIT WORK;
   DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
      || ASSERT_EQUALS(salidael,salida_Esperada_el));

      EXCEPTION
      WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
        || ASSERT_EQUALS(false,salida_Esperada_el));
        ROLLBACK;
    END ELIMINAR;
/*Copiar y pegar el procedimiento actualizar*/

  PROCEDURE ACTUALIZAR (
    nombre_prueba_up VARCHAR2,
    id_oferta IN OFERTAS.CODIGO%TYPE,
    fechaini IN OFERTAS.FECHA_INICIO%TYPE,
    fechafin IN OFERTAS.FECHA_FIN%TYPE,
    precioferta IN OFERTAS.PRECIOOFERTADO%TYPE,
    codigopro IN OFERTAS.CODIGO_PROD2 %TYPE,
    precioanterior IN OFERTAS.PRECIOANTES%TYPE,
    salida_esperada_up BOOLEAN
    )
  AS salidaup BOOLEAN := TRUE;
    oferta OFERTAS.CODIGO%TYPE;
    idoferta OFERTAS.CODIGO%TYPE;
```

```
BEGIN

    /*Llamada al procedimiento a probar*/


UPDATE_OFERTA(id_OFERTA,fechaini,fechafin,precioferta,codig
opro,precioanterior);

    idoferta := SEC_OFERTA.CURRVAL;

    /*Comprobar que se ha realizado con exito */

    SELECT CODIGO INTO oferta FROM OFERTAS WHERE
CODIGO = id_oferta;

    IF(idoferta <> oferta ) THEN

    salidaup := FALSE;

    END IF;

    COMMIT WORK;



  /*Mostrar resultados de la prueba*/

    DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '

    || ASSERT_EQUALS(salidaup,salida_Esperada_up));


    EXCEPTION
```

```
        WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
         || ASSERT_EQUALS(false,salida_Esperada_up));
         ROLLBACK;
        END ACTUALIZAR;

    end OFERTAS_PRUEBA;
/
-------------------------------------------------
-- Pruebas de ALBARANES
-------------------------------------------------

create or replace PACKAGE ALBARANES_PRUEBA IS
/*Creamos el procedimiento inicializar la cabecera*/

 Procedure INICIALIZAR;
 /*Creamos el procedimiento crear la cabecera*/
```

```
   /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE INSERTAR (

   nombre_prueba_in VARCHAR2,

   id_pedido IN ALBARANES.IDPEDIDO2%TYPE,

   id_factura IN ALBARANES.IDFACTURA1%TYPE,

   salida_esperada_in BOOLEAN

   );

/*Creamos el procedimiento borrar la cabecera*/

   /* Formato procedimientos:


nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE ELIMINAR (

   nombre_prueba_el VARCHAR2,

   idborrar IN ALBARANES.IDALBARAN%TYPE,

   salida_esperada_el BOOLEAN

   );

/*Creamos el procedimiento actualizar la cabecera*/
```

```
/* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE ACTUALIZAR (
   nombre_prueba_up VARCHAR2,
   id_albaranup IN ALBARANES.IDALBARAN%TYPE,
   id_pedidoup IN ALBARANES.IDPEDIDO2%TYPE,
   id_facturaUP IN ALBARANES.IDFACTURA1%TYPE,
   salida_esperada_up BOOLEAN
   );

END ALBARANES_PRUEBA;

/

create or replace PACKAGE BODY ALBARANES_PRUEBA AS
 procedure INICIALIZAR is
 begin
```

```
DELETE FROM ALBARANES;

end INICIALIZAR;



PROCEDURE INSERTAR (
  nombre_prueba_in VARCHAR2,
  id_pedido IN ALBARANES.IDPEDIDO2%TYPE,
  id_factura IN ALBARANES.IDFACTURA1%TYPE,
  salida_esperada_in BOOLEAN
  )
AS salidain BOOLEAN := TRUE;
  id_albaran ALBARANES.IDALBARAN%TYPE;
  albaran ALBARANES.IDPEDIDO2%TYPE;
BEGIN
    CREAR_ALBARAN(id_pedido, id_factura);
    id_albaran := SEC_ALBARAN.CURRVAL;
    SELECT ALBARANES.IDPEDIDO2 INTO albaran FROM
ALBARANES WHERE IDALBARAN = id_albaran;
    IF(albaran<>id_pedido ) THEN
```

```
    salidain := FALSE;
    END IF;
    COMMIT WORK;
  /*Mostrar resultados de la prueba*/
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
    || ASSERT_EQUALS(salidain,salida_Esperada_in));

    EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
     || ASSERT_EQUALS(false,salida_Esperada_in));
     ROLLBACK;
    END INSERTAR;

 /*Copiar y pegar el procedimiento borrar*/
  PROCEDURE ELIMINAR (
  nombre_prueba_el VARCHAR2,
  idborrar IN ALBARANES.IDALBARAN%TYPE,
  salida_esperada_el BOOLEAN
```

```
    )
AS salidael BOOLEAN := TRUE;
    numeroAlbaranes NUMBER;
BEGIN
    BORRAR_ALBARAN(idborrar);
    SELECT COUNT(*) INTO numeroAlbaranes FROM
ALBARANES
    WHERE IDALBARAN = idborrar;
    IF(numeroAlbaranes <> 0) THEN
    salidael:= FALSE;
    END IF;
    COMMIT WORK;
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
      || ASSERT_EQUALS(salidael,salida_Esperada_el));

     EXCEPTION
     WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
      || ASSERT_EQUALS(false,salida_Esperada_el));
```

```
        ROLLBACK;
    END ELIMINAR;
  /*Copiar y pegar el procedimiento actualizar*/

  PROCEDURE ACTUALIZAR (
    nombre_prueba_up VARCHAR2,
    id_albaranup IN ALBARANES.IDALBARAN%TYPE,
    id_pedidoup IN ALBARANES.IDPEDIDO2%TYPE,
    id_facturaUP IN ALBARANES.IDFACTURA1%TYPE,
    salida_Esperada_up BOOLEAN
    )
 AS salidaup BOOLEAN := TRUE;
    id_albaran2 ALBARANES.IDALBARAN%TYPE;
    albaran ALBARANES.IDPEDIDO2%TYPE;

 BEGIN
       /*Llamada al procedimiento a probar*/
       UPDATE_ALBARAN(id_albaranup, id_pedidoup,
id_facturaUP);
```

```
id_albaran2 := SEC_ALBARAN.CURRVAL;

/*Comprobar que se ha realizado con exito */

SELECT ALBARANES.IDPEDIDO2 INTO albaran FROM
ALBARANES WHERE IDALBARAN = id_albaran2;

IF(albaran <> id_pedidoup ) THEN

salidaup := FALSE;

END IF;

COMMIT WORK;



/*Mostrar resultados de la prueba*/

DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '

|| ASSERT_EQUALS(salidaup,salida_Esperada_up));


EXCEPTION

WHEN OTHERS THEN

 DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '

 || ASSERT_EQUALS(false,salida_Esperada_up));

 ROLLBACK;
```

```
      END ACTUALIZAR;


   end ALBARANES_PRUEBA;


/

-------------------------------------------------

-- Pruebas de DIRECCIONES

-------------------------------------------------


create or replace PACKAGE DIRECCIONES_PRUEBA IS

/*Creamos el procedimiento inicializar la cabecera*/


 Procedure INICIALIZAR;

 /*Creamos el procedimiento crear la cabecera*/

   /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE INSERTAR (

   nombre_prueba_in VARCHAR2,
```

```
    ciudad IN direcciones.CIUDAD%TYPE,
    numero IN direcciones.NUMERO%TYPE,
    cp IN direcciones.CP%TYPE,
    calle IN direcciones.CALLE%TYPE,
  salida_esperada_in BOOLEAN
  );
/*Creamos el procedimiento borrar la cabecera*/
  /* Formato procedimientos:

nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/
 PROCEDURE ELIMINAR (
  nombre_prueba_el VARCHAR2,
  id_direccionborrar IN DIRECCIONES.ID_DIRECCION%TYPE,
  salida_esperada_el BOOLEAN
  );
/*Creamos el procedimiento actualizar la cabecera*/
/* Formato procedimientos:
```

```
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE ACTUALIZAR (

   nombre_prueba_up VARCHAR2,

   id_direccionup IN DIRECCIONES.ID_DIRECCION%TYPE,

   ciudadup IN direcciones.CIUDAD%TYPE,

   cpup IN direcciones.CP%TYPE,

   calleup IN direcciones.CALLE%TYPE,

   numeroup IN direcciones.NUMERO%TYPE,

   salida_esperada_up BOOLEAN

   );


END DIRECCIONES_PRUEBA;


/


create or replace PACKAGE BODY DIRECCIONES_PRUEBA AS

 procedure INICIALIZAR is
```

```
begin

/*Se borran todas las tablas*/


DELETE FROM DIRECCIONES;

end INICIALIZAR;


/*Copiar y pegar el procedimiento crear*/


PROCEDURE INSERTAR (

  nombre_prueba_in VARCHAR2,

   ciudad IN direcciones.CIUDAD%TYPE,

   numero IN direcciones.NUMERO%TYPE,

   cp IN direcciones.CP%TYPE,

   calle IN direcciones.CALLE%TYPE,

  salida_esperada_in BOOLEAN

  )



AS salidain BOOLEAN := TRUE;
```

```
    id_direccioni DIRECCIONES.ID_DIRECCION%TYPE;
    direccion DIRECCIONES.CALLE%TYPE;
BEGIN
    /*Llamada al procedimiento a probar*/
    NUEVA_DIRECCION(ciudad, numero, cp, calle);
    id_direccioni := SEC_DIRECCION.CURRVAL;
    /*Comprobar que se ha realizado con exito */
    SELECT DIRECCIONES.CALLE INTO direccion FROM
DIRECCIONES WHERE ID_DIRECCION = id_direccioni;
    IF(direccion<>calle ) THEN
    salidain := FALSE;
    END IF;
    COMMIT WORK;
  /*Mostrar resultados de la prueba*/
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
    || ASSERT_EQUALS(salidain,salida_Esperada_in));

    EXCEPTION
    WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '
        || ASSERT_EQUALS(false,salida_Esperada_in));
        ROLLBACK;
       END INSERTAR;


 /*Copiar y pegar el procedimiento borrar*/
   PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   id_direccionborrar IN DIRECCIONES.ID_DIRECCION%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numeroDirecciones NUMBER;
BEGIN
   ELIMINA_DIRECCION(id_direccionborrar);
   SELECT COUNT(*) INTO numeroDirecciones FROM
DIRECCIONES
   WHERE ID_DIRECCION = id_direccionborrar;
   IF(numeroDirecciones <> 0) THEN
```

```
salidael:= FALSE;
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
   || ASSERT_EQUALS(salidael,salida_Esperada_el));

   EXCEPTION
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
    || ASSERT_EQUALS(false,salida_Esperada_el));
    ROLLBACK;
END ELIMINAR;
/*Copiar y pegar el procedimiento actualizar*/

PROCEDURE ACTUALIZAR (
nombre_prueba_up VARCHAR2,
id_direccionup IN DIRECCIONES.ID_DIRECCION%TYPE,
ciudadup IN direcciones.CIUDAD%TYPE,
cpup IN direcciones.CP%TYPE,
```

```
   calleup IN direcciones.CALLE%TYPE,
   numeroup IN direcciones.NUMERO%TYPE,
   salida_esperada_up BOOLEAN
   )
AS salidaup BOOLEAN := TRUE;
   id_direccion DIRECCIONES.ID_DIRECCION%TYPE;
   direccion DIRECCIONES.CALLE%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/

UPDATE_DIRECCION(id_direccionup,ciudadup,cpup,calleup,numeroup);
      id_direccion := SEC_DIRECCION.CURRVAL;
      /*Comprobar que se ha realizado con exito */
      SELECT DIRECCIONES.CALLE INTO direccion FROM DIRECCIONES WHERE ID_DIRECCION = id_direccion;
      IF(direccion <> calleup ) THEN
      salidaup := FALSE;
      END IF;
      COMMIT WORK;
```

```
/*Mostrar resultados de la prueba*/
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
    || ASSERT_EQUALS(salidaup,salida_Esperada_up));

    EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
     || ASSERT_EQUALS(false,salida_Esperada_up));
     ROLLBACK;
    END ACTUALIZAR;

  end DIRECCIONES_PRUEBA;

/

 ------------------------------------------------

-- Pruebas de CLIENTES

-------------------------------------------------
```

```
create or replace PACKAGE CLIENTES_PRUEBA IS

/*Creamos el procedimiento inicializar la cabecera*/


 Procedure INICIALIZAR;

/*Creamos el procedimiento crear la cabecera*/

   /* Formato procedimientos:
nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE INSERTAR (

   nombre_prueba_in VARCHAR2,

    nombreCLI IN clientes.NOMBRE_CLI%TYPE,

    appCLI IN clientes.APP_CLI%TYPE,

    apmCLI IN clientes.apm_CLI%TYPE,

    emailCLI IN clientes.EMAIL%TYPE,

    dniCLI IN clientes.DNI%TYPE,

    telefonoCLI IN clientes.telefono%TYPE,

    IDDIRECCION IN clientes.ID_DIRECCION2%TYPE,

   salida_esperada_in BOOLEAN

   );
```

```
/*Creamos el procedimiento borrar la cabecera*/

  /* Formato procedimientos:


nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE ELIMINAR (

   nombre_prueba_el VARCHAR2,

   IDCLIENTEborrar IN CLIENTES.IDCLIENTE%TYPE,

   salida_esperada_el BOOLEAN

   );

/*Creamos el procedimiento actualizar la cabecera*/

/* Formato procedimientos:


nombre_procedimiento(numbre_prueba,valores_entrada,salida_esperada)*/

 PROCEDURE ACTUALIZAR (

   nombre_prueba_up VARCHAR2,

   idclienteUP IN CLIENTES.IDCLIENTE%TYPE,

    nombreUP IN CLIENTES.NOMBRE_CLI%TYPE,

    appUP IN CLIENTES.APP_CLI%TYPE,
```

```
    apmUP IN CLIENTES.APM_CLI%TYPE,
    emailUP IN CLIENTES.EMAIL%TYPE,
    dniUP IN CLIENTES.DNI%TYPE,
    telefonoUP IN CLIENTES.telefono%TYPE,
    IDDIRECCIONUP IN CLIENTES.ID_DIRECCION2%TYPE,
   salida_esperada_up BOOLEAN
   );

END CLIENTES_PRUEBA;

/

create or replace PACKAGE BODY CLIENTES_PRUEBA AS
 procedure INICIALIZAR is
 begin
 /*Se borran todas las tablas*/

 DELETE FROM CLIENTES;
 end INICIALIZAR;
```

```
/*Copiar y pegar el procedimiento crear*/


PROCEDURE INSERTAR (
  nombre_prueba_in VARCHAR2,
   nombreCLI IN clientes.NOMBRE_CLI%TYPE,
   appCLI IN clientes.APP_CLI%TYPE,
   apmCLI IN clientes.apm_CLI%TYPE,
   emailCLI IN clientes.EMAIL%TYPE,
   dniCLI IN clientes.DNI%TYPE,
   telefonoCLI IN clientes.telefono%TYPE,
   IDDIRECCION IN clientes.ID_DIRECCION2%TYPE,
  salida_esperada_in BOOLEAN
  )
AS salidain BOOLEAN := TRUE;
  id_CLIENTE CLIENTES.IDCLIENTE%TYPE;
  CLIENTE CLIENTES.DNI%TYPE;
BEGIN
    /*Llamada al procedimiento a probar*/
```

```
CREAR_CLIENTE(nombreCLI,appcli,apmCLI,emailCLI,dniCLI,tel
efonoCLI,IDDIRECCION);

    id_CLIENTE := SEC_CLIENTE.CURRVAL;

    /*Comprobar que se ha realizado con exito */

    SELECT CLIENTES.DNI INTO CLIENTE FROM CLIENTES
WHERE IDCLIENTE = id_CLIENTE;

    IF(cliente<>dniCLI ) THEN

    salidain := FALSE;

    END IF;

    COMMIT WORK;

  /*Mostrar resultados de la prueba*/

    DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '

    || ASSERT_EQUALS(salidain,salida_Esperada_in));


    EXCEPTION

    WHEN OTHERS THEN

     DBMS_OUTPUT.PUT_LINE(nombre_prueba_in || ': '

     || ASSERT_EQUALS(false,salida_Esperada_in));

     ROLLBACK;
```

```
    END INSERTAR;


 /*Copiar y pegar el procedimiento borrar*/
   PROCEDURE ELIMINAR (
   nombre_prueba_el VARCHAR2,
   IDCLIENTEborrar IN CLIENTES.IDCLIENTE%TYPE,
   salida_esperada_el BOOLEAN
   )
AS salidael BOOLEAN := TRUE;
   numeroCLIENTES NUMBER;
BEGIN
   ELIMINAR_CLIENTE(IDCLIENTEborrar);
   SELECT COUNT(*) INTO numeroCLIENTES FROM CLIENTES
   WHERE IDCLIENTE = IDCLIENTEborrar;
   IF(numeroCLIENTES <> 0) THEN
   salidael:= FALSE;
   END IF;
   COMMIT WORK;
   DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '
```

```
    || ASSERT_EQUALS(salidael,salida_Esperada_el));


   EXCEPTION

   WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE(nombre_prueba_el || ': '

    || ASSERT_EQUALS(false,salida_Esperada_el));

    ROLLBACK;

 END ELIMINAR;

/*Copiar y pegar el procedimiento actualizar*/


PROCEDURE ACTUALIZAR (

 nombre_prueba_up VARCHAR2,

 idclienteUP IN CLIENTES.IDCLIENTE%TYPE,

  nombreUP IN clientes.NOMBRE_CLI%TYPE,

  appUP IN clientes.APP_CLI%TYPE,

  apmUP IN clientes.APM_CLI%TYPE,

  emailUP IN clientes.EMAIL%TYPE,

  dniUP IN clientes.DNI%TYPE,

  telefonoUP IN clientes.telefono%TYPE,
```

```
    IDDIRECCIONUP IN clientes.ID_DIRECCION2%TYPE,
   salida_esperada_up BOOLEAN
   )
AS salidaup BOOLEAN := TRUE;
   id_CLIENTE CLIENTES.IDCLIENTE%TYPE;
   CLIENTE CLIENTES.DNI%TYPE;
BEGIN
      /*Llamada al procedimiento a probar*/
      UPDATE_CLIENTE(idclienteUP,
nombreUP,appUP,apmUP,emailUP,dniUP,telefonoUP,IDDIRECCI
ONUP);
      id_CLIENTE := SEC_CLIENTE.CURRVAL;
      /*Comprobar que se ha realizado con exito */
 SELECT CLIENTES.DNI INTO CLIENTE FROM CLIENTES
WHERE IDCLIENTE = id_CLIENTE;
      IF(cliente<>dniUP ) THEN
      salidaup := FALSE;
      END IF;
      COMMIT WORK;
```

```
/*Mostrar resultados de la prueba*/
    DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
    || ASSERT_EQUALS(salidaup,salida_Esperada_up));

    EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE(nombre_prueba_up || ': '
     || ASSERT_EQUALS(false,salida_Esperada_up));
     ROLLBACK;
    END ACTUALIZAR;

  end CLIENTES_PRUEBA;

/
```

Fichero Pruebas

```
-----------------------------------------------
-- PRUEBAS ALMACENES
-----------------------------------------------
```

```
create or replace PACKAGE ALMACENES_PRUEBA IS

/*Creamos el procedimiento inicializar la cabecera*/

 Procedure inicializar;

/*Creamos el procedimiento crear la cabecera*/

 PROCEDURE CREAR_ALMACEN (
   nombreal IN ALMACENES.NOMBRE%TYPE,
   id_direccional IN ALMACENES.ID_DIRECCION3%TYPE
   );
/*Creamos el procedimiento borrar la cabecera*/

 PROCEDURE BORRAR_ALMACEN (
   id_almacendel IN ALMACENES.IDALMACEN%TYPE
   );
/*Creamos el procedimiento actualizar la cabecera*/
```

```
PROCEDURE UPDATE_ALMACEN (
  id_almacenup IN ALMACENES.IDALMACEN%TYPE,
  nombreup IN ALMACENES.NOMBRE%TYPE,
  id_direccionup IN ALMACENES.ID_DIRECCION3%TYPE
  );

END ALMACENES_PRUEBA;
/
create or replace PACKAGE BODY ALMACENES_PRUEBA AS
 procedure inicializar is
 begin
 /*Se borran todas las tablas*/

 DELETE FROM almacenes;
 end inicializar;

 /*Copiar y pegar el procedimiento crear*/

PROCEDURE CREAR_ALMACEN (
```

```
   nombreal IN ALMACENES.NOMBRE%TYPE,
   id_direccional IN ALMACENES.ID_DIRECCION3%TYPE
   )
IS idalmacen ALMACENES.IDALMACEN%TYPE;

BEGIN
      SELECT sec_almacen.NEXTVAL INTO idalmacen FROM dual;
      INSERT INTO almacenes (IDALMACEN, NOMBRE,
ID_DIRECCION3)
      VALUES(idalmacen, nombreal, id_direccional);
      DBMS_OUTPUT.PUT_LINE(nombreal);
END CREAR_ALMACEN;
 /*Copiar y pegar el procedimiento borrar*/

   PROCEDURE BORRAR_ALMACEN (
   id_almacendel IN ALMACENES.IDALMACEN%TYPE
   )
IS idalmacen ALMACENES.IDALMACEN%TYPE;
```

```
BEGIN
   DELETE FROM almacenes WHERE IDALMACEN = idalmacen;
END BORRAR_ALMACEN;
 /*Copiar y pegar el procedimiento actualizar*/

 PROCEDURE UPDATE_ALMACEN (
   id_almacenup IN ALMACENES.IDALMACEN%TYPE,
   nombreup IN ALMACENES.NOMBRE%TYPE,
   id_direccionup IN ALMACENES.ID_DIRECCION3%TYPE
   )
IS idalmacen ALMACENES.IDALMACEN%TYPE;

BEGIN
    UPDATE almacenes
    SET IDALMACEN = id_almacenup,
    NOMBRE = nombreup,
    ID_DIRECCION3 = id_direccionup
    WHERE IDALMACEN = id_almacenup;
END UPDATE_ALMACEN;
```

```
   end almacenes_prueba;

/

-----------------------------------------------

--  PRUEBAS ALBARANES

-----------------------------------------------

create or replace PACKAGE ALBARANES_PRUEBA IS

/*Creamos el procedimiento inicializar la cabecera*/

 Procedure inicializar;

 /*Creamos el procedimiento crear la cabecera*/

 PROCEDURE CREAR_ALBARAN (

      id_pedido IN ALBARANES.IDPEDIDO2%TYPE

   );
```

```
/*Creamos el procedimiento borrar la cabecera*/


PROCEDURE BORRAR_ALBARAN (
  idborrar IN ALBARANES.IDALBARAN%TYPE
  );
/*Creamos el procedimiento actualizar la cabecera*/


PROCEDURE UPDATE_ALBARAN (
     id_albaranup IN ALBARANES.IDALBARAN%TYPE,
     id_pedidoup IN ALBARANES.IDPEDIDO2%TYPE
  );


END ALBARANES_PRUEBA;
/
create or replace PACKAGE BODY ALBARANES_PRUEBA AS
 procedure inicializar is
 begin
 /*Se borran todas las tablas*/
```

```
DELETE FROM ALBARANES;

end inicializar;


/*Copiar y pegar el procedimiento crear*/


PROCEDURE CREAR_ALBARAN (

    id_pedido IN ALBARANES.IDPEDIDO2%TYPE

  )

IS id_albaran ALBARANES.IDALBARAN%TYPE;

  BEGIN

    SELECT sec_albaran.NEXTVAL INTO id_albaran FROM dual;

    INSERT INTO albaranes (IDALBARAN, IDPEDIDO2)

    VALUES(id_albaran, id_pedido);

END CREAR_ALBARAN;

/*Copiar y pegar el procedimiento borrar*/


  PROCEDURE BORRAR_ALBARAN (

  idborrar IN ALBARANES.IDALBARAN%TYPE

  )
```

```
IS idalbaran ALBARANES.IDALBARAN%TYPE;


BEGIN
   DELETE FROM albaranes WHERE IDALBARAN = idborrar;
END BORRAR_ALBARAN;
 /*Copiar y pegar el procedimiento actualizar*/


 PROCEDURE UPDATE_ALBARAN (
     id_albaranup IN ALBARANES.IDALBARAN%TYPE,
     id_pedidoup IN ALBARANES.IDPEDIDO2%TYPE
   )
IS
BEGIN
    UPDATE albaranes
    SET IDPEDIDO2 = id_pedidoup
    WHERE IDALBARAN = id_albaranup;
END UPDATE_ALBARAN;


  end ALBARANES_PRUEBA;
```

```
/

----------------------------------------------------

--  PRUEBAS CLIENTES

----------------------------------------------------

create or replace PACKAGE CIENTES_PRUEBA IS


/*Creamos el procedimiento inicializar la cabecera*/


 Procedure inicializar;


 /*Creamos el procedimiento crear la cabecera*/


 PROCEDURE CREAR_CLIENTE (

   nombreCLI IN CLIENTES.NOMBRE_CLI%TYPE,

   app_cli IN CLIENTES.APP_CLI%TYPE,

   apm_cli IN CLIENTES.APM_CLI%TYPE,

   email_cli IN CLIENTES.EMAIL%TYPE,

   dni_cli IN CLIENTES.DNI%TYPE,
```

```
  tel_cli IN CLIENTES.TELEFONO%TYPE,
  id_direccion_CLI IN CLIENTES.ID_DIRECCION2%TYPE
  );
/*Creamos el procedimiento borrar la cabecera*/

PROCEDURE ELIMINAR_CLIENTE (
  id_clienteDEL IN CLIENTES.IDCLIENTE%TYPE
  );
/*Creamos el procedimiento actualizar la cabecera*/

PROCEDURE UPDATE_CLIENTE (
  id_clienteUP IN CLIENTES.IDCLIENTE%TYPE,
  nombreUP IN CLIENTES.NOMBRE_CLI%TYPE,
  appUP IN CLIENTES.APP_CLI%TYPE,
  apmUP IN CLIENTES.APM_CLI%TYPE,
  emaUP IN CLIENTES.EMAIL%TYPE,
  dniUP IN CLIENTES.DNI%TYPE,
  telUP IN CLIENTES.TELEFONO%TYPE,
  id_direccion_CLIUP IN CLIENTES.ID_DIRECCION2%TYPE
```

```
    );


END CIENTES_PRUEBA;

/

create or replace PACKAGE BODY CIENTES_PRUEBA AS

 procedure inicializar is

 begin

 /*Se borran todas las tablas*/


 DELETE FROM CLIENTES;

 end inicializar;


 /*Copiar y pegar el procedimiento crear*/


 PROCEDURE CREAR_CLIENTE (

   nombreCLI IN CLIENTES.NOMBRE_CLI%TYPE,

   app_cli IN CLIENTES.APP_CLI%TYPE,

   apm_cli IN CLIENTES.APM_CLI%TYPE,

   email_cli IN CLIENTES.EMAIL%TYPE,
```

```
   dni_cli IN CLIENTES.DNI%TYPE,
   tel_cli IN CLIENTES.TELEFONO%TYPE,
   id_direccion_CLI IN CLIENTES.ID_DIRECCION2%TYPE
   )
IS id_cliente CLIENTES.IDCLIENTE%TYPE;

BEGIN
      SELECT sec_cliente.NEXTVAL INTO id_cliente FROM dual;
      INSERT INTO clientes (IDCLIENTE, NOMBRE_CLI,
APP_CLI, APM_CLI, EMAIL, DNI, TELEFONO,
ID_DIRECCION2)
      VALUES(id_cliente, nombreCLI, app_cli, apm_cli, email_cli,
dni_cli, tel_cli, id_direccion_CLI);
END CREAR_CLIENTE;
 /*Copiar y pegar el procedimiento borrar*/

   PROCEDURE ELIMINAR_CLIENTE (
   id_clienteDEL IN CLIENTES.IDCLIENTE%TYPE
   )
IS id_cliente CLIENTES.IDCLIENTE%TYPE;
```

```
BEGIN
    DELETE FROM clientes WHERE IDCLIENTE = id_clienteDEL;
END ELIMINAR_CLIENTE;
 /*Copiar y pegar el procedimiento actualizar*/

 PROCEDURE UPDATE_CLIENTE (
  id_clienteUP IN CLIENTES.IDCLIENTE%TYPE,
  nombreUP IN CLIENTES.NOMBRE_CLI%TYPE,
  appUP IN CLIENTES.APP_CLI%TYPE,
  apmUP IN CLIENTES.APM_CLI%TYPE,
  emaUP IN CLIENTES.EMAIL%TYPE,
  dniUP IN CLIENTES.DNI%TYPE,
  telUP IN CLIENTES.TELEFONO%TYPE,
  id_direccion_CLIUP IN CLIENTES.ID_DIRECCION2%TYPE
  )
IS
BEGIN
   UPDATE clientes
```

```
    SET NOMBRE_CLI = nombreUP,
    APP_CLI = appUP,
    APM_CLI = apmUP,
    EMAIL = emaUP,
    DNI = dniUP,
    TELEFONO = telUP,
    ID_DIRECCION2 = id_direccion_CLIUP
    WHERE IDCLIENTE = id_clienteUP;
END UPDATE_CLIENTE;

  end CIENTES_PRUEBA;

/

----------------------------------------------
--  PRUEBAS DIRECCIONES
----------------------------------------------

create or replace PACKAGE DIRECCIONES_PRUEBA IS
```

```
/*Creamos el procedimiento inicializar la cabecera*/

Procedure inicializar;

/*Creamos el procedimiento crear la cabecera*/

PROCEDURE NUEVA_DIRECCION (
    ciudad IN direcciones.CIUDAD%TYPE,
    numero IN direcciones.NUMERO%TYPE,
    cp IN direcciones.CP%TYPE,
    calle IN direcciones.CALLE%TYPE);
/*Creamos el procedimiento borrar la cabecera*/

PROCEDURE ELIMINA_DIRECCION(
      id_direccionborrar IN
DIRECCIONES.ID_DIRECCION%TYPE
);
/*Creamos el procedimiento actualizar la cabecera*/
```

```
PROCEDURE UPDATE_DIRECCION
(     id_direccionup IN DIRECCIONES.ID_DIRECCION%TYPE,
      ciudadup IN direcciones.CIUDAD%TYPE,
      cpup IN direcciones.CP%TYPE,
      calleup IN direcciones.CALLE%TYPE,
      numeroup IN direcciones.NUMERO%TYPE);

END DIRECCIONES_PRUEBA;
/
create or replace PACKAGE BODY DIRECCIONES_PRUEBA AS
 procedure inicializar is
 begin
 /*Se borran todas las tablas*/

 DELETE FROM DIRECCIONES;
 end inicializar;

 /*Copiar y pegar el procedimiento crear*/
```

```
 PROCEDURE NUEVA_DIRECCION

( ciudad IN direcciones.CIUDAD%TYPE, numero IN
direcciones.NUMERO%TYPE, cp IN direcciones.CP%TYPE, calle
IN direcciones.CALLE%TYPE)

AS id_direccion direcciones.ID_DIRECCION%TYPE;


BEGIN

    SELECT sec_direccion.NEXTVAL INTO id_direccion FROM
dual;

    INSERT INTO direcciones (ID_DIRECCION, CIUDAD, CP,
CALLE, NUMERO)

    VALUES(id_direccion, ciudad, cp, calle, numero);

END NUEVA_DIRECCION;

 /*Copiar y pegar el procedimiento borrar*/


  PROCEDURE ELIMINA_DIRECCION(

    id_direccionborrar IN
DIRECCIONES.ID_DIRECCION%TYPE

)
```

```
IS id_direccion direcciones.ID_DIRECCION%TYPE;


BEGIN

   DELETE FROM direcciones WHERE id_direccion =
id_direccionborrar;

END ELIMINA_DIRECCION;

 /*Copiar y pegar el procedimiento actualizar*/


 PROCEDURE UPDATE_DIRECCION

(      id_direccionup IN DIRECCIONES.ID_DIRECCION%TYPE,

       ciudadup IN direcciones.CIUDAD%TYPE,

       cpup IN direcciones.CP%TYPE,

       calleup IN direcciones.CALLE%TYPE,

       numeroup IN direcciones.NUMERO%TYPE)

IS

BEGIN

    UPDATE direcciones

    SET ciudad = ciudadup,

    cp = cpup,
```

```
    calle = calleup,
    numero = numeroup
    WHERE ID_DIRECCION = id_direccionup;
END UPDATE_DIRECCION;

  end DIRECCIONES_PRUEBA;

/
-----------------------------------------------
-- PRUEBAS ESTADO DE PEDIDOS
-----------------------------------------------

create or replace PACKAGE ESTADOPEDIDOS_PRUEBA IS

/*Creamos el procedimiento inicializar la cabecera*/
PROCEDURE inicializar;

PROCEDURE CREAR_ESTADOPEDIDO (
      estado IN ESTADOPEDIDOS.ESTADOPEDIDO%TYPE,
```

```
      pedido IN ESTADOPEDIDOS.IDPEDIDO1%TYPE
   );
/*Creamos el procedimiento borrar la cabecera*/

 PROCEDURE BORRAR_ESTADOPEDIDO (
   idborrar IN ESTADOPEDIDOS.IDPEDIDO1%TYPE
   );

 PROCEDURE UPDATE_ESTADOPEDIDO (
      pedido IN ESTADOPEDIDOS.IDPEDIDO1%TYPE,
     estado IN ESTADOPEDIDOS.ESTADOPEDIDO%TYPE
   );

END ESTADOPEDIDOS_PRUEBA;
/
create or replace PACKAGE BODY ESTADOPEDIDOS_PRUEBA
AS
procedure inicializar is
 begin
```

```
DELETE FROM ESTADOPEDIDOS;

end inicializar;


/*Copiar y pegar el procedimiento actualizar*/

  PROCEDURE CREAR_ESTADOPEDIDO (

    estado IN ESTADOPEDIDOS.ESTADOPEDIDO%TYPE,

    pedido IN ESTADOPEDIDOS.IDPEDIDO1%TYPE

  )

IS codigo ESTADOPEDIDOS.ESTADOPEDIDO%TYPE;

 BEGIN

    INSERT INTO ESTADOPEDIDOS (ESTADOPEDIDO,
IDPEDIDO1)

    VALUES(estado, pedido);

END CREAR_ESTADOPEDIDO;


  PROCEDURE BORRAR_ESTADOPEDIDO (

  idborrar IN ESTADOPEDIDOS.IDPEDIDO1%TYPE

  )

IS
```

```
BEGIN

   DELETE FROM ESTADOPEDIDOS WHERE IDPEDIDO1 =
idborrar;

END BORRAR_ESTADOPEDIDO;


 PROCEDURE UPDATE_ESTADOPEDIDO (

      pedido IN ESTADOPEDIDOS.IDPEDIDO1%TYPE,

     estado IN ESTADOPEDIDOS.ESTADOPEDIDO%TYPE

   )

IS

BEGIN

   UPDATE ESTADOPEDIDOS

   SET ESTADOPEDIDO = estado

   WHERE IDPEDIDO1 = pedido;

END UPDATE_ESTADOPEDIDO;


   end ESTADOPEDIDOS_PRUEBA;

/

----------------------------------------------
```

```
-- PRUEBAS LINEAS DE PEDIDOS

-------------------------------------------------

create or replace PACKAGE LINEAPEDIDOS_PRUEBA AS

    /*Creamos el procedimiento inicializar la cabecera*/


 PROCEDURE inicializar;

/*Creamos el procedimiento crear la cabecera*/


 PROCEDURE CREAR_LINEAPEDIDO (

      cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,

      precio IN LINEAPEDIDOS.PRECIO%TYPE,

      producto IN LINEAPEDIDOS.CODIGO_PROD1%TYPE,

      pedido IN LINEAPEDIDOS.IDPEDIDO3%TYPE

   );

/*Creamos el procedimiento borrar la cabecera*/


 PROCEDURE ELIMINA_LINEAPEDIDO (

   num IN LINEAPEDIDOS.NUMLINE%TYPE

   );
```

```
 /*Creamos el procedimiento actualizar la cabecera*/
 PROCEDURE UPDATE_LINEAPEDIDO (
      num IN LINEAPEDIDOS.NUMLINE%TYPE,
      cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,
      precio IN LINEAPEDIDOS.PRECIO%TYPE,
      idProduc IN LINEAPEDIDOS.CODIGO_PROD1%TYPE
   );

END LINEAPEDIDOS_PRUEBA;
/
create or replace PACKAGE BODY LINEAPEDIDOS_PRUEBA AS
/*Se borran todas las tablas*/
 procedure inicializar is
 begin
 DELETE FROM LINEAPEDIDOS;
 end inicializar;
 /*Copiar y pegar el procedimiento crear*/
PROCEDURE CREAR_LINEAPEDIDO (
      cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,
```

```
    precio IN LINEAPEDIDOS.PRECIO%TYPE,
    producto IN LINEAPEDIDOS.CODIGO_PROD1%TYPE,
    pedido IN LINEAPEDIDOS.IDPEDIDO3%TYPE
  )
IS codigo LINEAPEDIDOS.NUMLINE%TYPE;
  BEGIN
    SELECT SEC_LINEAPEDIDOS.NEXTVAL INTO codigo
FROM dual;
    INSERT INTO LINEAPEDIDOS (NUMLINE,
CANTIDADPEDIDO, PRECIO, CODIGO_PROD1, IDPEDIDO3)
    VALUES(codigo, cant, precio, producto, pedido);
END CREAR_LINEAPEDIDO;
/*Copiar y pegar el procedimiento borrar*/
PROCEDURE ELIMINA_LINEAPEDIDO (
    num IN LINEAPEDIDOS.NUMLINE%TYPE
  )
IS NUMLINE LINEAPEDIDOS.NUMLINE%TYPE;
  BEGIN
  DELETE FROM LINEAPEDIDOS WHERE NUMLINE = num;
```

```
END ELIMINA_LINEAPEDIDO;

/*Copiar y pegar el procedimiento actualizar*/

PROCEDURE UPDATE_LINEAPEDIDO (

      num IN LINEAPEDIDOS.NUMLINE%TYPE,

      cant IN LINEAPEDIDOS.CANTIDADPEDIDO%TYPE,

      precio IN LINEAPEDIDOS.PRECIO%TYPE,

      idProduc IN LINEAPEDIDOS.CODIGO_PROD1%TYPE

   )

IS

   BEGIN

    UPDATE LINEAPEDIDOS

    SET CANTIDADPEDIDO = cant,

    PRECIO = precio

    WHERE NUMLINE = num;

END UPDATE_LINEAPEDIDO;

   end LINEAPEDIDOS_PRUEBA;


/

-----------------------------------------------
```

```
-- PRUEBAS PEDIDOS

-----------------------------------------------

create or replace PACKAGE PEDIDOS_PRUEBA IS


/*Creamos el procedimiento inicializar la cabecera*/


 Procedure inicializar;


 /*Creamos el procedimiento crear la cabecera*/


 PROCEDURE CREAR_PEDIDO (
      subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,
      cliente IN PEDIDOS.IDCLIENTE1%TYPE,
      direccion IN PEDIDOS.ID_DIRECCION1%TYPE
   );
 /*Creamos el procedimiento borrar la cabecera*/


 PROCEDURE ELIMINA_PEDIDO (
   codigoborrar IN PEDIDOS.IDPEDIDO%TYPE
```

```
    );
/*Creamos el procedimiento actualizar la cabecera*/

PROCEDURE UPDATE_PEDIDO (
    codigopedido IN PEDIDOS.IDPEDIDO%TYPE,
        fech IN PEDIDOS.FECHA%TYPE,
    subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,
    cliente IN PEDIDOS.IDCLIENTE1%TYPE,
    direccion IN PEDIDOS.ID_DIRECCION1%TYPE
    );

END PEDIDOS_PRUEBA;
/
create or replace PACKAGE BODY PEDIDOS_PRUEBA AS
 procedure inicializar is
 begin
 /*Se borran todas las tablas*/

 DELETE FROM PEDIDOS;
```

```
end inicializar;


/*Copiar y pegar el procedimiento crear*/


PROCEDURE CREAR_PEDIDO (
    subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,
    cliente IN PEDIDOS.IDCLIENTE1%TYPE,
    direccion IN PEDIDOS.ID_DIRECCION1%TYPE
  )
AS codigo PEDIDOS.IDPEDIDO%TYPE;
  fecha PEDIDOS.FECHA%TYPE;
  BEGIN
    SELECT SEC_PEDIDO.NEXTVAL INTO codigo FROM dual;
    SELECT TO_CHAR(SysDate,'DD/MM/YYYY HH24:MI:SS')
todays_date INTO fecha FROM dual;
    INSERT INTO PEDIDOS (IDPEDIDO, FECHA,
PRECIOTOTAL, IDCLIENTE1, ID_DIRECCION1)
    VALUES(codigo, fecha, subtotal, cliente, direccion);
    INSERT INTO ESTADOPEDIDOS
(ESTADOPEDIDO,IDPEDIDO1)
```

```
        VALUES ('En Proceso', codigo);
END CREAR_PEDIDO;
 /*Copiar y pegar el procedimiento borrar*/

   PROCEDURE ELIMINA_PEDIDO (
   codigoborrar IN PEDIDOS.IDPEDIDO%TYPE
)
IS codigo PEDIDOS.IDPEDIDO%TYPE;

BEGIN
   DELETE FROM PEDIDOS WHERE IDPEDIDO = codigoborrar;
   DELETE FROM ESTADOPEDIDOS WHERE IDPEDIDO1 =
codigoborrar;
   DELETE FROM LINEAPEDIDOS WHERE IDPEDIDO3 =
codigoborrar;
END ELIMINA_PEDIDO;
 /*Copiar y pegar el procedimiento actualizar*/

  PROCEDURE UPDATE_PEDIDO (
```

```
        codigopedido IN PEDIDOS.IDPEDIDO%TYPE,
              fech IN PEDIDOS.FECHA%TYPE,
        subtotal IN PEDIDOS.PRECIOTOTAL%TYPE,
        cliente IN PEDIDOS.IDCLIENTE1%TYPE,
        direccion IN PEDIDOS.ID_DIRECCION1%TYPE
    )
IS
    BEGIN
     UPDATE PEDIDOS
     SET PRECIOTOTAL = subtotal,
    ID_DIRECCION1 = direccion
     WHERE IDPEDIDO = codigopedido;
END UPDATE_PEDIDO;

    end pedidos_prueba;

/
----------------------------------------------
-- PRUEBAS PRODUCTOS
```

```
----------------------------------------------------

create or replace PACKAGE PRODUCTOS_PRUEBA IS


/*Creamos el procedimiento inicializar la cabecera*/


 Procedure inicializar;


/*Creamos el procedimiento crear la cabecera*/


 PROCEDURE CREAR_PRODUCTO (
      nombre IN productos.NOMBRE%TYPE,
      descripcion IN productos.DESCRIPCION%TYPE,
      tipoproducto IN productos.TIPOPRODUCTO%TYPE,
      disponibilidad IN productos.DISPONIBILIDAD%TYPE,
      precio IN productos.PRECIO%TYPE,
      urldocu IN productos.URLDOCU%TYPE,
      urlimg IN productos.URLIMG%TYPE,
      stock IN productos.STOCK%TYPE,
      partnumber IN productos.PARTNUMBER%TYPE,
```

```
    modelo IN productos.MODELO%TYPE
  );
/*Creamos el procedimiento borrar la cabecera*/

PROCEDURE ELIMINA_PRODUCTO (
  codigoborrar IN productos.CODIGO%TYPE
  );
/*Creamos el procedimiento actualizar la cabecera*/

PROCEDURE UPDATE_PRODUCTO (
    codigoup IN productos.CODIGO%TYPE,
    nombreup IN productos.NOMBRE%TYPE,
    descripcionup IN productos.DESCRIPCION%TYPE,
    tipoproductoup IN productos.TIPOPRODUCTO%TYPE,
    disponibilidadup IN productos.DISPONIBILIDAD%TYPE,
    precioup IN productos.PRECIO%TYPE,
    urldocuup IN productos.URLDOCU%TYPE,
    urlimgup IN productos.URLIMG%TYPE,
    stockup IN productos.STOCK%TYPE,
```

```
        partnumberup IN productos.PARTNUMBER%TYPE,
        modeloup IN productos.MODELO%TYPE
    );

END PRODUCTOS_PRUEBA;
/
create or replace PACKAGE BODY PRODUCTOS_PRUEBA AS
 procedure inicializar is
 begin
 /*Se borran todas las tablas*/

 DELETE FROM PRODUCTOS;
 end inicializar;

 /*Copiar y pegar el procedimiento crear*/

 PROCEDURE CREAR_PRODUCTO (
        nombre IN productos.NOMBRE%TYPE,
        descripcion IN productos.DESCRIPCION%TYPE,
```

```
        tipoproducto IN productos.TIPOPRODUCTO%TYPE,
        disponibilidad IN productos.DISPONIBILIDAD%TYPE,
        precio IN productos.PRECIO%TYPE,
        urldocu IN productos.URLDOCU%TYPE,
        urlimg IN productos.URLIMG%TYPE,
        stock IN productos.STOCK%TYPE,
        partnumber IN productos.PARTNUMBER%TYPE,
        modelo IN productos.MODELO%TYPE
    )
IS codigo productos.CODIGO%TYPE;
    BEGIN
        SELECT sec_producto.NEXTVAL INTO codigo FROM dual;
        INSERT INTO productos (CODIGO, NOMBRE,
DESCRIPCION, TIPOPRODUCTO, DISPONIBILIDAD, PRECIO,
URLDOCU, URLIMG, STOCK, PARTNUMBER, MODELO)
        VALUES(codigo, nombre, descripcion, tipoproducto,
disponibilidad, precio, urldocu, urlimg, stock, partnumber,
modelo);
END CREAR_PRODUCTO;
 /*Copiar y pegar el procedimiento borrar*/
```

```
   PROCEDURE ELIMINA_PRODUCTO (
   codigoborrar IN productos.CODIGO%TYPE
)
IS codigo productos.CODIGO%TYPE;

BEGIN
   DELETE FROM productos WHERE codigo = codigoborrar;
END ELIMINA_PRODUCTO;
 /*Copiar y pegar el procedimiento actualizar*/

 PROCEDURE UPDATE_PRODUCTO (
            codigoup IN productos.CODIGO%TYPE,
      nombreup IN productos.NOMBRE%TYPE,
      descripcionup IN productos.DESCRIPCION%TYPE,
      tipoproductoup IN productos.TIPOPRODUCTO%TYPE,
      disponibilidadup IN productos.DISPONIBILIDAD%TYPE,
      precioup IN productos.PRECIO%TYPE,
      urldocuup IN productos.URLDOCU%TYPE,
```

```
    urlimgup IN productos.URLIMG%TYPE,
    stockup IN productos.STOCK%TYPE,
    partnumberup IN productos.PARTNUMBER%TYPE,
    modeloup IN productos.MODELO%TYPE
  )
IS
  BEGIN
   UPDATE productos
   SET nombre = nombreup,
   descripcion = descripcionup,
   tipoproducto = tipoproductoup,
   disponibilidad = disponibilidadup,
   precio = precioup,
   urldocu = urldocuup,
   urlimg = urlimgup,
   stock = stockup,
   partnumber = partnumberup,
   modelo = modeloup
   WHERE codigo = codigoup;
```

```
END UPDATE_PRODUCTO;

   end PRODUCTOS_prueba;

/

----------------------------------------------
--  PRUEBAS OFERTAS
----------------------------------------------
create or replace PACKAGE OFERTAS_PRUEBA IS

/*Creamos el procedimiento inicializar la cabecera*/

 Procedure inicializar;

 /*Creamos el procedimiento crear la cabecera*/

 PROCEDURE CREAR_OFERTA (
      fechaini IN OFERTAS.FECHA_INICIO%TYPE,
```

```
    fechafin IN OFERTAS.FECHA_FIN%TYPE,
    precioDes IN OFERTAS.PRECIOOFERTADO%TYPE,
    codigoPro IN OFERTAS.CODIGO%TYPE
  );
/*Creamos el procedimiento borrar la cabecera*/

PROCEDURE ELIMINA_OFERTA (
  id_ofertaborrar IN OFERTAS.CODIGO%TYPE
  );
/*Creamos el procedimiento actualizar la cabecera*/

PROCEDURE UPDATE_OFERTA (
    id_oferta IN OFERTAS.CODIGO%TYPE,
    fechaini IN OFERTAS.FECHA_INICIO%TYPE,
    fechafin IN OFERTAS.FECHA_FIN%TYPE,
    precioferta IN OFERTAS.PRECIOOFERTADO%TYPE,
    codigopro IN OFERTAS.CODIGO_PROD2 %TYPE,
    precioanterior IN OFERTAS.PRECIOANTES%TYPE
  );
```

```
END OFERTAS_PRUEBA;

/

create or replace PACKAGE BODY OFERTAS_PRUEBA AS

 procedure inicializar is

 begin

 /*Se borran todas las tablas*/


 DELETE FROM OFERTAS;

 end inicializar;


 /*Copiar y pegar el procedimiento crear*/


 PROCEDURE CREAR_OFERTA (

     fechaini IN OFERTAS.FECHA_INICIO%TYPE,

     fechafin IN OFERTAS.FECHA_FIN%TYPE,

     precioDes IN OFERTAS.PRECIOOFERTADO%TYPE,

     codigoPro IN OFERTAS.CODIGO%TYPE

   )
```

```
IS idOferta OFERTAS.CODIGO%TYPE;
   precioanterior PRODUCTOS.PRECIO%TYPE;
   BEGIN
      SELECT SEC_OFERTA.NEXTVAL INTO idOferta FROM dual;
      SELECT PRODUCTOS.PRECIO INTO precioanterior FROM
PRODUCTOS
      WHERE PRODUCTOS.CODIGO = codigoPro;
      INSERT INTO OFERTAS (CODIGO,
FECHA_INICIO,FECHA_FIN,PRECIOOFERTADO,CODIGO_PRO
D2,PRECIOANTES)
      VALUES(idOferta,
fechaini,fechafin,precioDes,codigoPro,precioanterior);
END CREAR_OFERTA;
 /*Copiar y pegar el procedimiento borrar*/

   PROCEDURE ELIMINA_OFERTA (
      id_ofertaborrar IN OFERTAS.CODIGO%TYPE
)
IS id_oferta direcciones.ID_DIRECCION%TYPE;
   restaurarprecio PRODUCTOS.PRECIO%TYPE;
```

```
   idproducto OFERTAS.CODIGO_PROD2%TYPE;
BEGIN
   SELECT OFERTAS.PRECIOANTES INTO restaurarprecio
FROM OFERTAS
   WHERE CODIGO = id_ofertaborrar;
   SELECT OFERTAS.CODIGO_PROD2 INTO idproducto FROM
OFERTAS
   WHERE CODIGO = id_ofertaborrar;
   UPDATE PRODUCTOS
   SET PRECIO = restaurarprecio
   WHERE PRODUCTOS.CODIGO = idproducto;
   DELETE FROM OFERTAS WHERE id_oferta = id_ofertaborrar;
END ELIMINA_OFERTA;
 /*Copiar y pegar el procedimiento actualizar*/

 PROCEDURE UPDATE_OFERTA
(      id_oferta IN OFERTAS.CODIGO%TYPE,
       fechaini IN OFERTAS.FECHA_INICIO%TYPE,
       fechafin IN OFERTAS.FECHA_FIN%TYPE,
```

```
        precioferta IN OFERTAS.PRECIOOFERTADO%TYPE,
        codigopro IN OFERTAS.CODIGO_PROD2 %TYPE,
        precioanterior IN OFERTAS.PRECIOANTES%TYPE)
IS
BEGIN
    UPDATE OFERTAS
    SET FECHA_INICIO = fechaini,
    FECHA_FIN = fechafin,
    PRECIOOFERTADO = precioferta,
    CODIGO_PROD2 = codigopro,
   PRECIOANTES = precioanterior
   WHERE CODIGO = id_oferta;
END UPDATE_OFERTA;

   end OFERTAS_prueba;

/
----------------------------------------------
-- PRUEBAS FACTURAS
```

```
----------------------------------------------------

create or replace PACKAGE FACTURAS_PRUEBA IS


/*Creamos el procedimiento inicializar la cabecera*/


 Procedure inicializar;

 /*Creamos el procedimiento crear la cabecera*/

 PROCEDURE CREAR_FACTURA (

      albaran IN FACTURAS.IDALBARAN1%TYPE,

      pago IN FACTURAS.TIPOFORMAPAGO%TYPE

   );

/*Creamos el procedimiento borrar la cabecera*/


 PROCEDURE BORRAR_FACTURA (

   idborrar IN FACTURAS.IDFACTURA%TYPE

   );

/*Creamos el procedimiento actualizar la cabecera*/


 PROCEDURE UPDATE_FACTURA (
```

```
    id_factura IN FACTURAS.IDFACTURA%TYPE,
    precio IN FACTURAS.PRECIOTOTAL%TYPE,
    pago IN FACTURAS.TIPOFORMAPAGO%TYPE
  );

END FACTURAS_PRUEBA;
/
create or replace PACKAGE BODY FACTURAS_PRUEBA AS
 procedure inicializar is
 begin
 /*Se borran todas las tablas*/

 DELETE FROM FACTURAS;
 end inicializar;

 /*Copiar y pegar el procedimiento crear*/

 PROCEDURE CREAR_FACTURA (
    albaran IN FACTURAS.IDALBARAN1%TYPE,
```

```
    pago IN FACTURAS.TIPOFORMAPAGO%TYPE

  )

IS factura FACTURAS.IDFACTURA%TYPE;

  precio PEDIDOS.PRECIOTOTAL%TYPE;

  BEGIN

    SELECT SEC_FACTURA.NEXTVAL INTO factura FROM
dual;

    SELECT PEDIDOS.PRECIOTOTAL INTO precio FROM
ALBARANES,PEDIDOS where ALBARANES.IDPEDIDO2 =
PEDIDOS.IDPEDIDO;

    INSERT INTO FACTURAS (IDFACTURA,
TIPOFORMAPAGO,PRECIOTOTAL,IDALBARAN1)

    VALUES(factura, pago, precio,albaran);

END CREAR_FACTURA;

 /*Copiar y pegar el procedimiento borrar*/


  PROCEDURE BORRAR_FACTURA (

  idborrar IN FACTURAS.IDFACTURA%TYPE

  )

IS factura FACTURAS.IDFACTURA%TYPE;
```

```
BEGIN
   DELETE FROM FACTURAS WHERE IDFACTURA = idborrar;
END BORRAR_FACTURA;
 /*Copiar y pegar el procedimiento actualizar*/

 PROCEDURE UPDATE_FACTURA (
     id_factura IN FACTURAS.IDFACTURA%TYPE,
     precio IN FACTURAS.PRECIOTOTAL%TYPE,
     pago IN FACTURAS.TIPOFORMAPAGO%TYPE

   )
IS
BEGIN
    UPDATE FACTURAS
    SET PRECIOTOTAL = precio,
     TIPOFORMAPAGO = pago
    WHERE IDFACTURA = id_factura;
END UPDATE_FACTURA;
```

```
   end FACTURAS_PRUEBA;


/
CURSORES

--Cursor productos por precio Mayor a Menor


DECLARE

    CURSOR cur_productos IS

    SELECT nombre, precio, tipoproducto, disponibilidad FROM
productos

    ORDER BY precio DESC;

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Productos ordenados por
precio.');

        FOR fila IN cur_productos LOOP

          DBMS_OUTPUT.PUT_LINE(fila.nombre|| ' '
||fila.precio|| '€ ' ||fila.tipoproducto|| ' ' ||fila.disponibilidad);

        END LOOP;

    END;
```

```
--Cursor productos por precio Menor a Mayor




DECLARE

    CURSOR cur_productos IS

    SELECT nombre, precio, tipoproducto, disponibilidad FROM
productos

    ORDER BY precio ASC;

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Productos ordenados por
precio.');

      FOR fila IN cur_productos LOOP

        DBMS_OUTPUT.PUT_LINE(fila.nombre|| ' '
||fila.precio|| '€ ' ||fila.tipoproducto|| ' ' ||fila.disponibilidad);

      END LOOP;

    END;



--Cursor productos por tipo producto
```

```
DECLARE
    CURSOR cur_productos IS
    SELECT nombre, precio, tipoproducto, disponibilidad FROM
productos
    ORDER BY precio ASC;
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Productos ordenados por tipo de
producto.');
      FOR fila IN cur_productos LOOP
        DBMS_OUTPUT.PUT_LINE(fila.nombre|| ' '
||fila.precio|| '€ ' ||fila.tipoproducto|| ' ' ||fila.disponibilidad);
      END LOOP;
    END;
```

```
--Cursor productos por disponibilidad

DECLARE
    CURSOR cur_productos IS
```

```
SELECT nombre, precio, tipoproducto, disponibilidad FROM
productos

ORDER BY disponibilidad;

BEGIN

DBMS_OUTPUT.PUT_LINE('Productos ordenados por
disponibilidad.');

FOR fila IN cur_productos LOOP

DBMS_OUTPUT.PUT_LINE(fila.nombre|| ' '
||fila.precio|| '€ ' ||fila.tipoproducto|| ' ' ||fila.disponibilidad);

END LOOP;

END;


--Cursor Albaranes por id pedido


DECLARE

CURSOR cur_albaran IS

SELECT idalbaran, idpedido2, idfactura1 FROM albaranes

ORDER BY idpedido2;

BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('Albaranes ordenados por id
pedido.');

      FOR fila IN cur_albaran LOOP

        DBMS_OUTPUT.PUT_LINE(fila.idalbaran|| ''
||fila.idpedido2|| '' ||fila.idfactura1);

      END LOOP;

    END;


--Cursor Albaranes por id factura


DECLARE

    CURSOR cur_albaran IS

    SELECT idalbaran, idpedido2, idfactura1 FROM albaranes

    ORDER BY idfactura1;

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Albaranes ordenados por id
factura.');

      FOR fila IN cur_albaran LOOP

        DBMS_OUTPUT.PUT_LINE(fila.idalbaran|| ''
||fila.idpedido2|| '' ||fila.idfactura1);
```

```
        END LOOP;

    END;


--Cursor Almacenes por nombre


DECLARE

    CURSOR cur_almacen IS

    SELECT idalmacen, nombre, ID_DIRECCION3 FROM
almacenes

    ORDER BY nombre;

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Almacenes ordenados por
nombre.');

        FOR fila IN cur_almacen LOOP

          DBMS_OUTPUT.PUT_LINE(fila.idalmacen||' '
||fila.nombre||' '||fila.ID_DIRECCION3);

        END LOOP;

    END;


--Cursor Clientes por DNI
```

```
DECLARE
    CURSOR cur_cliente IS
    SELECT idcliente, nombre_cli, dni FROM clientes
    ORDER BY dni;
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Clientes ordenados por dni.');
      FOR fila IN cur_cliente LOOP
        DBMS_OUTPUT.PUT_LINE(fila.idcliente||' '
||fila.nombre_cli||' '||fila.dni);
      END LOOP;
    END;


--Cursor Clientes por Nombre


DECLARE
    CURSOR cur_cliente IS
    SELECT idcliente, nombre_cli, dni FROM clientes
    ORDER BY nombre_cli;
```

```
BEGIN

    DBMS_OUTPUT.PUT_LINE('Clientes ordenados por
nombre.');

        FOR fila IN cur_cliente LOOP

            DBMS_OUTPUT.PUT_LINE(fila.idcliente|| ' '
||fila.nombre_cli|| ' ' ||fila.dni);

        END LOOP;

    END;


--Cursor Direcciones por Ciudad


DECLARE

    CURSOR cur_direccion IS

    SELECT id_direccion, ciudad, calle FROM direcciones

    ORDER BY ciudad;

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Direcciones ordenados por
ciudad.');

        FOR fila IN cur_direccion LOOP
```

```
            DBMS_OUTPUT.PUT_LINE(fila.id_direccion||' '
||fila.ciudad||' '||fila.calle);

        END LOOP;

    END;


--Cursor Pedidos por fecha DESC

DECLARE

    CURSOR cur_pedido IS

    SELECT idpedido, fecha, idcliente1, preciototal FROM pedidos

    ORDER BY fecha DESC;

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Pedidos ordenados por fecha.');

        FOR fila IN cur_pedido LOOP

            DBMS_OUTPUT.PUT_LINE(fila.idpedido||' '
||fila.fecha||' '||fila.preciototal||'€ '||fila.idcliente1);

        END LOOP;

    END;

--Cursor Pedidos por fecha ASC

DECLARE
```

```
    CURSOR cur_pedido IS
    SELECT idpedido, fecha, idcliente1, preciototal FROM pedidos
    ORDER BY fecha ASC;
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Pedidos ordenados por fecha.');
      FOR fila IN cur_pedido LOOP
        DBMS_OUTPUT.PUT_LINE(fila.idpedido|| ' '
||fila.fecha|| ' ' ||fila.preciototal|| '€ ' ||fila.idcliente1);
      END LOOP;
    END;
--Cursor Pedidos por id_Cliente
DECLARE
    CURSOR cur_pedido IS
    SELECT idpedido, fecha, idcliente1, preciototal FROM pedidos
    ORDER BY idcliente1;
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Pedidos ordenados por fecha.');
      FOR fila IN cur_pedido LOOP
```

```
            DBMS_OUTPUT.PUT_LINE(fila.idpedido||' '
||fila.fecha||' '||fila.preciototal||'€ '||fila.idcliente1);

        END LOOP;

    END;
```

ANEXO:

Se realizo una reunión con uno de los dueños de la empresa y con la secretaria de dirección, tuvo lugar el 13/10/2017 a las 12:30 y finalizo a las 14:30.
Se dialogo sobre los requisitos del proyecto, los objetivos y las necesidades del cliente.

Se trataron 3 temas barajaron 3 posibilidades:

1. Implementar la tienda en la web.
2. Implementar la gestión de Taller.
3. Implementar la gestión de la plantilla en la web.

Por prioridad y sugerencia de la dirección se decidió que el objetivo de este proyecto sea la implementación de la tienda web, ya que la gestión de la plantilla se consideró innecesaria por estar compuesta esta de tan solo 20 empleados, finalmente la gestión del taller se declinó por ser menor su prioridad para las necesidades de la empresa.

Una vez enfocado el proyecto se trató de dar forma a este, tratamos sobre los productos que se incluirán en la tienda, si se realizaran ofertas y promociones o no, la forma de pago, envío y facturación. Esto se nos aclaró vía e-mail más tarde.

Transcripción del e-mail recibido de Susana Manzano Guerra secretaria de dirección, el día 18/10/2017:

En principio vamos a empezar a vender:

1. Equipos de soldadura: entre 50 y 60 modelos.
2. Consumibles: sobre 200 tipos diferentes, entre hilos, varillas y electrodos y además de cada uno los hay en diámetros diferentes.
3. Repuestos: se incluiría accesorios varios de soldadura: pantallas, manómetros, guantes, torchas, válvulas…. quizás otros 200 artículos.
4. Ofertas varias.

La forma de pago será por adelantado: con tarjeta de crédito, PayPal o transferencia bancaria. Otra cosa son los portes que según el material, cantidad o importe se le sumarán lo que corresponda o serán gratis.

ACTA DE REUNIÓN

Comienza la sesión a las 12:30 horas del 13 de octubre de 2017, en las instalaciones de SAIND.

Comparecen:

- Don Francisco Javier Moral Moral, dueño y Director.
- Doña Susana Manzano Guerra, secretaria de Dirección.
- José Julián Manzano Montenegro, encargado del proyecto.

Se propone las siguientes opciones:

1° Implementación tienda Web, Don Francisco.
2° Implementación gestión de Taller, José Julián.
3° Implementación gestión de la plantilla, Doña Susana.

Se Dialoga sobre las propuestas, declinando la 3° opción por considerar que la plantilla de 20 empleados no necesita gestión extra. Por otra parte, se considera de menor prioridad la gestión del taller de reparaciones, por tanto, se acuerda aceptar la propuesta de Don Francisco (1ª) Implementación tienda Web.

Finalmente, Don Francisco propone una serie de requisitos preliminares que se concretarán posteriormente, a falta de la aprobación de Don Juan Carlos Moral Moral, dueño y Director de SAIND:

- Formas de Pago.
- Lineas de productos en la tienda Web.
- Ofertas y Descuentos.
- Como tratar los gastos de portes si hubiera.
- Datos clientes y productos.
- Permisos y tipos de Usuarios.

Finaliza la sesión, a las 14:30 horas del 13 de Octubre del 2017, en las instalaciones de SAZNO

Firmado:

Don Francisco Javier Moral Moral.
Doña Susana Manzano Guerra.
José Julián Manzano Montenegro.