

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 3**
2. **Glosario términos 4**
3. **Visión general del sistema 9**
4. **Catálogo de requisitos 12**
5. **Pruebas de aceptación 18**
6. **Mapa conceptual 20**
7. **Matrices de Trazabilidad 22**
8. **Modelo Relacional en 3FN 23**
9. **Modelo Tecnológico 24**

**Anexo 23**

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.

1. **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.

1. **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.

1. **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-08 Información de almacén:**

Como director,

Quiero disponer de la información:

* Nombre.
* Dirección.

**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.

1. **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,

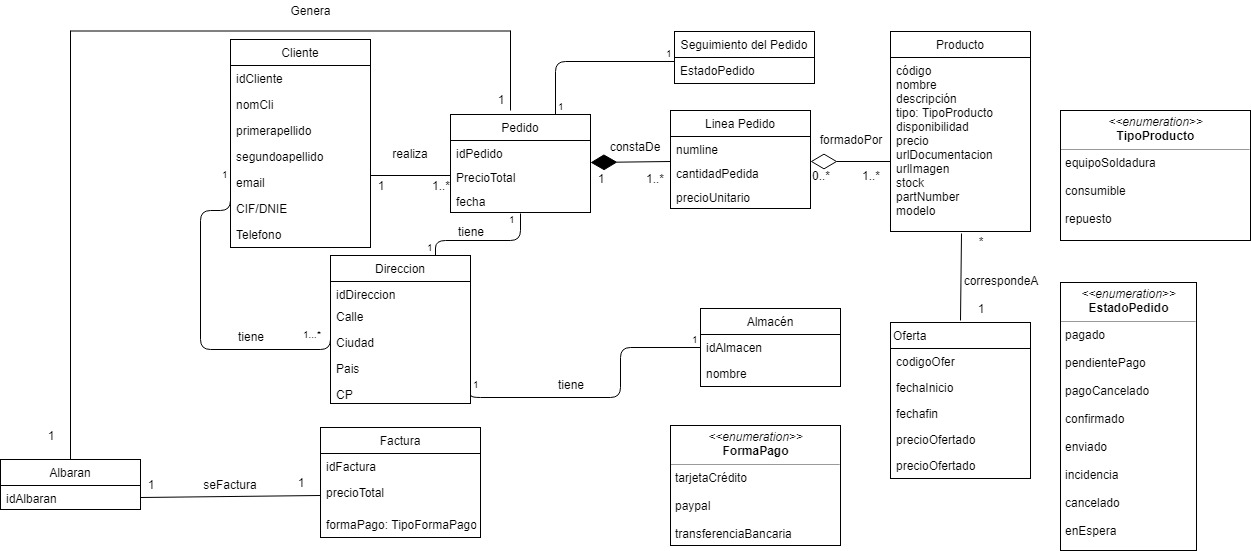
Quiero que antes de realizar él envió de un pedido, se compruebe que el pago no ha sido cancelado.

**Prueba de Aceptación:**

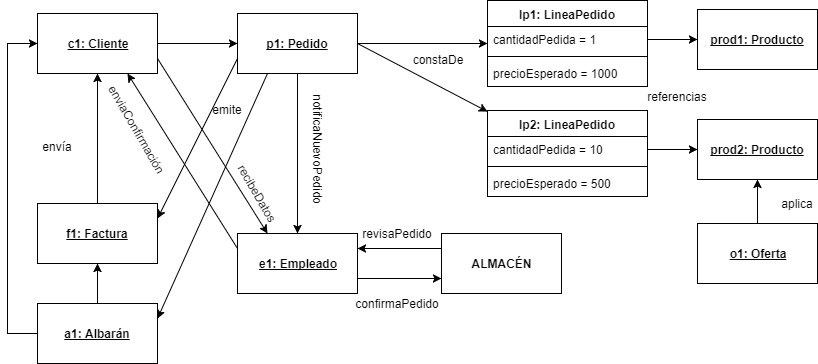
* El cliente realiza un pedido, se comprueba que el pago no ha sido cancelado y se envía.
* El cliente realiza un pedido, el pago se ha cancelado y no se envía.

1. **Modelo Conceptual.**

* **Diagrama UML.**

****

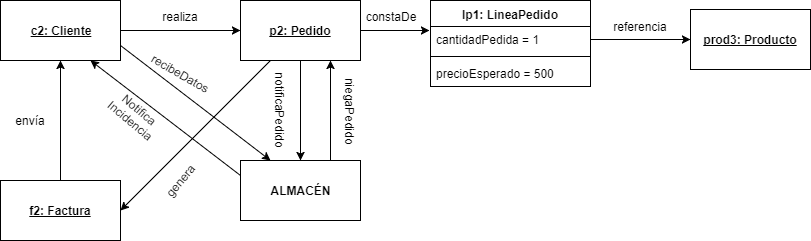
* **Escenario de prueba**

****

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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |  |  |  |  |  |  |  |  | ✓ | ✓ |  |  |  |  |  |  |  |  |  |

1. **Matrices de trazabilidad**
2. **Modelo relacional en 3FN**

**Imagen que contiene captura de pantalla

Descripción generada con confianza alta**

1. **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\_PROD2,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,disponibilidad,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,tipoproductoup,

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,codigopro,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,telefonoCLI,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,IDDIRECCIONUP);

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\_PROD2,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.

