INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE Y A LOS SISTEMAS DE INFORMACIÓN

ENTREGA 3

MOTOS CANDE

Candelario Luna, Luis

Morales Moreno, Sergio

Rodríguez Vidosa, Luis

Tutor: Alejandro Fernández-Montes González

VERSIÓN 3.0

ÍNDICE

1. INTRODUCCIÓN AL PROBLEMA

Motos Cande, situada en la ciudad de Zafra (Extremadura), es una empresa que tiene una doble actividad: venta de ropa de temática motorista y servicio técnico a motocicletas averiadas.

La empresa está situada en una nave industrial a las afueras de la ciudad, concretamente en el polígono industrial Los Caños y está formada por 3 miembros:

-Jose Antonio Candelario Montaño. Es el gerente de la empresa.

-Isidro Candelario Montaño. Es el dependiente de la empresa.

-Tomas Candelario Montaño. Es el encargado de la parte de servicio técnico.

Nuestros clientes desean crear una aplicación web con el fin de aumentar su número de ventas debido al decrecimiento de las mismas a causa del modelo de mercado actual. Como sabemos, en el modelo de mercado actual la mayoría de la competencia posee una tienda online influyendo fuertemente en las ventas de nuestro cliente. Además, el cliente presenta un problema relacionado con la gestión de citas ya que sus clientes desean ver y probar el vehículo que van a adquirir antes de comprarlo lo cual conlleva un sistema de gestión inexistente o bien desean acudir al servicio técnico para cualquier tipo de consulta relacionada con la reparación.

Como solución a este problema vamos a desarrollarle un sistema software mediante el cual pueda afrontar esta situación, pueda ponerse a la altura de su competencia e incluso gestione sus citas con el fin de solucionar este tipo de problemas.

2. GLOSARIO DE TÉRMINOS

Off-Road. - El término off-road hace referencia a una superficie donde es difícil transitar en vehículos comunes. La superficie puede ser tanto natural como artificial y está compuesta generalmente de arena, grava, y/o barro, en ocasiones cubierta de agua, nieve o hielo. Algunos de estos terrenos son tan ásperos que solamente pueden ser transitados en vehículos diseñados para este fin.

Enduro. - Nos encontramos ante una de las tres modalidades dentro del off-road. El Enduro es una modalidad del motociclismo que se practica en campo abierto.

Motocross. - Es una forma de competición off-road físicamente exigente que tiene lugar en todo tipo de condiciones celebrada en circuitos cerrados. Las motos de motocross se suelen caracterizar por su poco peso con el objetivo de proporcionar al piloto una posición una gran agilidad y rapidez.

Trail. - Es una forma de competición off-road que se realiza en pistas y caminos en buen estado. Las motos trail se suelen caracterizar por proporcionar al piloto una posición bastante alta y erguida que le permite un buen domino de la moto al mismo tiempo que confort y buena visibilidad. Por otra parte, suelen ser más pesadas, y menos ágiles que las motos pensadas únicamente para ser usadas en Motocross o Enduro.

Peto. - Protección que se utiliza en las modalidades de Motocross y Enduro que sirve para proteger la zona pectoral.

Mono. - Prenda de vestir de una sola pieza formada por un cuerpo semejante a una chaqueta y un pantalón unidos que suele utilizarse en el ámbito del motociclismo.

Casco abatible. - Es un casco formado por dos partes,se caracteriza porque la parte frontal que cubre la cara es abatible hacia arriba con el fin de lograr una mayor comodidad para el piloto.

Moto Custom. - Es un tipo de motocicleta que ofrece la posibilidad de personalización y modificación de acuerdo a los gustos del dueño de tal forma que la motocicleta acaba convirtiéndose en un reflejo de la personalidad y estética de la persona que la posee.

Cilindrada. - Es la denominación que se da a la suma del volumen útil de todos los cilindros de un motor.

3. VISIÓN GENERAL DEL SISTEMA

Nuestro sistema no solo ayudará a extender el mercado por internet sino también a mejorar el control sobre el stock en la tienda y el almacén, así como la organización de las ofertas y un seguimiento de qué artículos son los más vendidos. Además, gestionará las citas comentadas en la introducción del documento y ayudará por otra parte a saber cuándo hay que encargar más productos cuando queden fuera de stock.

Los objetivos del sistema para los distintos usuarios se definirán en la sección “4. Catálogo de requisitos”.

4. CATÁLOGO DE REQUISITOS

1. Requisitos generales/objetivos

A. Gestionar ventas

Como gerente,
quiero disponer de una aplicación web,
para ofrecer mis artículos y ofertas a clientes.

B. Gestionar pedidos

Como dependiente,
quiero poder llevar un control de los productos comprados a través de la aplicación web,
para poder distribuirlos.

C. Gestionar valoraciones

Como gerente,
quiero comprobar todas valoraciones recibidas,
para poder mejorar, de cara al futuro, los fallos de funcionamiento del negocio.

D. Gestionar citas

Como gerente,
quiero que mis clientes tengan disponibles un sistema de citas tanto como para una consulta como,
para evaluar un producto.

E. Adquirir productos

Como cliente,
quiero consultar los productos y ofertas disponibles del negocio,
para poder adquirir los productos que desee.

F. Solicitar citas

Como cliente,
quiero un sistema de solicitud de citas,
para poder ser atendido por el negocio de forma presencial.

2. Requisitos de información

RI-1. Información usuarios

Como gerente,
quiero disponer de la siguiente información sobre los usuarios:

-Email
-Contraseña
-Teléfono
-Tipo de usuario

Si el usuario es un cliente quiero disponer además de:

-Nombre
-Apellidos
-DNI
-Dirección

RI-2. Información pedidos

Como dependiente,
quiero disponer sobre la siguiente información sobre los pedidos:

-Número del pedido
-Fecha en la que se realiza el pedido
-Fecha de entrega del pedido
-Recogida o envío a domicilio
-Tipo de pago (Tarjeta, Transferencia)
-Cliente que realiza el pedido
-Estado del pedido (En espera, En tránsito, Entregado)
-Precio total del pedido
-Productos del pedido

RI-3. Información productos

Como gerente,
quiero disponer de la siguiente información sobre los productos:

-Código
-Nombre del producto
-Descripción
-Marca
-Tipo (Equipamiento, Motor, Recambio)
-Precio
-Oferta
-IVA
-Stock
-Stock Mínimo

Si el producto es un artículo de equipamiento quiero disponer además de:
-Talla
-Material
-Color

Si el producto tiene un motor quiero disponer además de:

-Cilindrada
-Estado (Nuevo, Seminuevo, Segunda Mano)
-Año de fabricación
-Tipo de producto (Motocicleta, maquinaria)
-Garantía

RI-4. Información citas

Como gerente,
quiero disponer de la siguiente información sobre las citas:

-Fecha acordada
-Asunto
-Descripción
-Tipo de cita (Tienda, Taller)
-Cliente correspondiente

RI-5. Información proveedor

Como gerente,
quiero disponer de la siguiente información sobre los proveedores:

-Código del proveedor
-Nombre de la compañía
-Email
-Teléfono
-Dirección
-Web

RI-6. Información valoración

Como gerente,
quiero disponer de la siguiente información sobre las valoraciones:

-Asunto
-Descripción
-Fecha de envío
-Cliente correspondiente

3. Reglas de negocio

RN-1. Máximo años garantía

Como gerente,
quiero que cada producto de tipo Motor tenga una garantía máxima de 6 años,
para tener un buen servicio postventa.

RN-2. Prohibición de cheque como método de pago

Como gerente,
quiero que no se puedan permitir compras mediante cheques,
para evitar fraudes.

RN-3. Devolución mediante tarjeta regalo

Como gerente,
quiero que las devoluciones de los productos no se hagan en metálico
para evitar el uso indebido de mis productos.

RN-4. Prohibición de compra

Como gerente,
quiero que no se puedan adquirir productos si no son abonados en el mismo momento de la compra,
para evitar pérdidas.

RN-5. Entrega de vehículo

Como gerente,
quiero que no se puedan retirar los productos de tipo motor sin haber abonado la factura o facturas correspondientes,
para evitar engaños.

4. Requisitos funcionales

RF-1. Gestionar pedido

Como dependiente,
quiero gestionar pedido de un cliente en cada momento,
para tener un seguimiento exhaustivo del mismo.

RF-2. Comprobar stock

Como gerente,
quiero recibir una notificación cuando un producto se esté agotando,
para reponerlo.

RF-3. Registro en el sistema

Como usuario,
quiero registrarme en el sistema,
para facilitar el pedido en futuros pedidos.

RF-4. Carrito de la compra

Como usuario,
quiero disponer de la opción de un carrito de la compra,
para poder organizar un pedido.

RF-5. Añadir/eliminar/editar productos

Como gerente,
quiero poder modificar el catálogo de la tienda online dependiendo de las necesidades de la tienda,
para ofrecer productos a la venta.

RF-6. Listar productos

Como cliente,
quiero que el sistema me ofrezca un conjunto de productos,
para poder seleccionarlos y comprarlos si lo deseo.

RF-7. Realizar pedido

Como usuario,
quiero poder realizar el pedido,
para adquirir producto que desee.

RF-8. Gestionar citas

Como gerente,
quiero poder gestionar las citas con los clientes,
para poder adecuarlas a mi horario.

RF-9: Visualizar valoraciones

Como gerente,
quiero poder visualizar las valoraciones de los clientes,
para poder mejorar mi negocio.

RF-10. Realizar valoraciones

Como cliente,
quiero poder realizar una valoración,
para así poder expresar mi opinión sobre el negocio.

RF-11. Solicitar citas

Como cliente,
quiero poder solicitar una cita,
para así poder contactar de forma presencial con la empresa.

RF-12. Consultar pedidos

Como usuario,
quiero poder ver el historial de pedidos que he efectuado,
para poder consultarlos si lo deseo.

RF-13: Consultar ofertas

Como usuario,
quiero poder ver los productos que estén en oferta,
para adquirirlos si lo deseo.

RF-14: Consultar contacto

Como usuario,
quiero poder consultar los datos de contacto de la empresa,
para ponerme en contacto con ellos si lo necesito.

RF-15: Consultar compras

Como gerente,
quiero poder almacenar todos los pedidos que se realicen
para poder consultarlas si lo deseo.

RF-16. Control exhaustivo stock

Como gerente,
quiero que haya un control exhaustivo de stock,
para que no se produzcan pedidos incorrectos o desabastecimiento en la tienda.

5. Requisitos no funcionales

RNF-1. Seguridad de contraseñas

Como usuario,
quiero que mis contraseñas sean almacenadas de forma encriptada,
para mejorar mi seguridad.

RNF-2. Rapidez

Como dependiente,
quiero que el software tenga poco tiempo de respuesta,
para que se gestionen los pedidos con mayor rapidez y se reciban lo antes posible notificaciones sobre el estado del envío con el fin de llevar un seguimiento adecuado del producto.

RNF-3. Acceso a la web

Como usuario,
quiero que la aplicación web esté disponible durante todo el año a cualquier hora,
para poder utilizar sus servicios cuando lo desee.

RNF-3. Idiomas

Como usuario,
quiero que la aplicación web esté disponible tanto en inglés como en español,
para poder internacionalizarla.

5. PRUEBAS DE ACEPTACIÓN

1. Pruebas de aceptación de Reglas de negocio

PA.1

- Un usuario intenta solucionar un fallo de fabricación de un producto de tipo motor cuya compra se realizó hace 6 años y el negocio se lo deniega.
- Un usuario intenta arreglar un fallo de fabricación de un producto de tipo motor cuya compra se realizó hace menos de 6 años y el negocio lo soluciona.

PA.2

- Un usuario intenta realizar una compra mediante un cheque y el negocio no acepta la compra.
- Un usuario intenta realizar una compra mediante cualquier otro método de pago permitido y el negocio permite la compra.

PA.3

- Un usuario realiza la devolución de un producto y el negocio le entrega una tarjeta regalo del mismo valor que el producto.

PA.4

- Un usuario intenta realizar una compra y abonar el importe correspondiente posteriormente y el negocio se lo impide.
- Un usuario realiza una compra y abona el importe correspondiente y el negocio permite la compra.

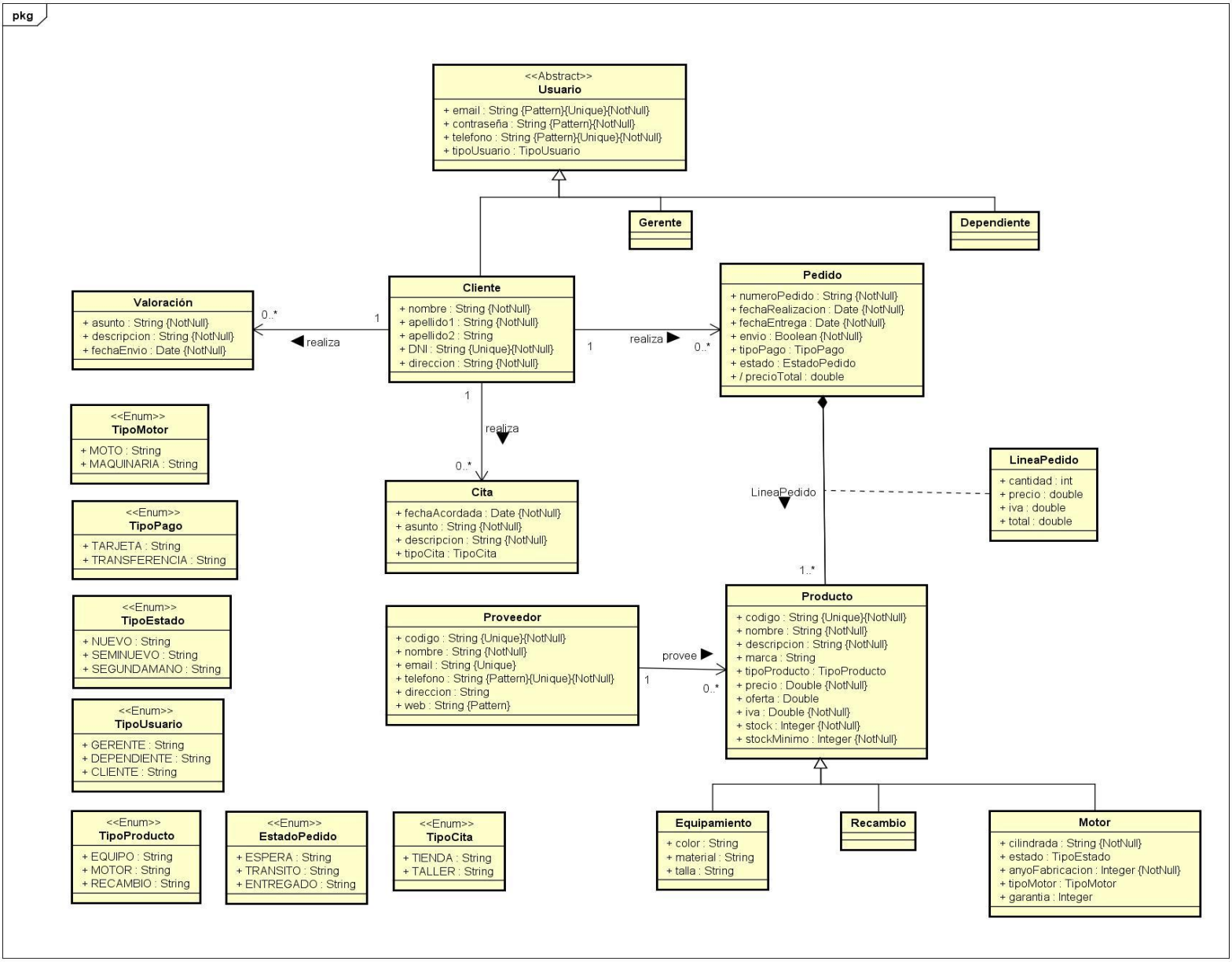
PA.5

- Un usuario intenta retirar su vehículo del taller sin haber abonado el importe correspondiente y el negocio se lo impide.

- Un usuario intenta retirar su vehículo del taller tras haber abonado el importe correspondiente y el negocio se lo permite.

6. MODELO CONCEPTUAL

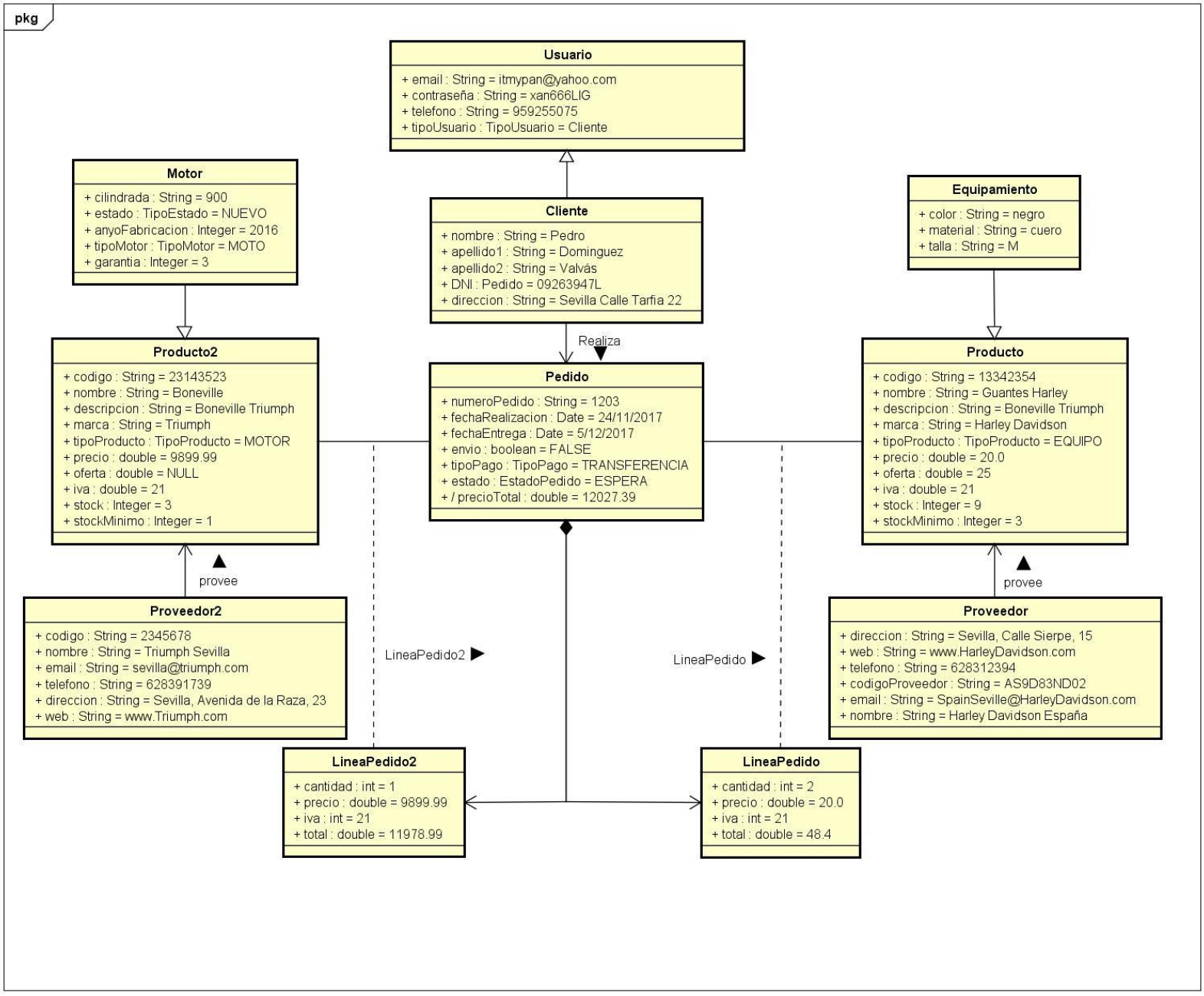
1. Diagrama de clases UML con restricciones



2. Escenarios de prueba

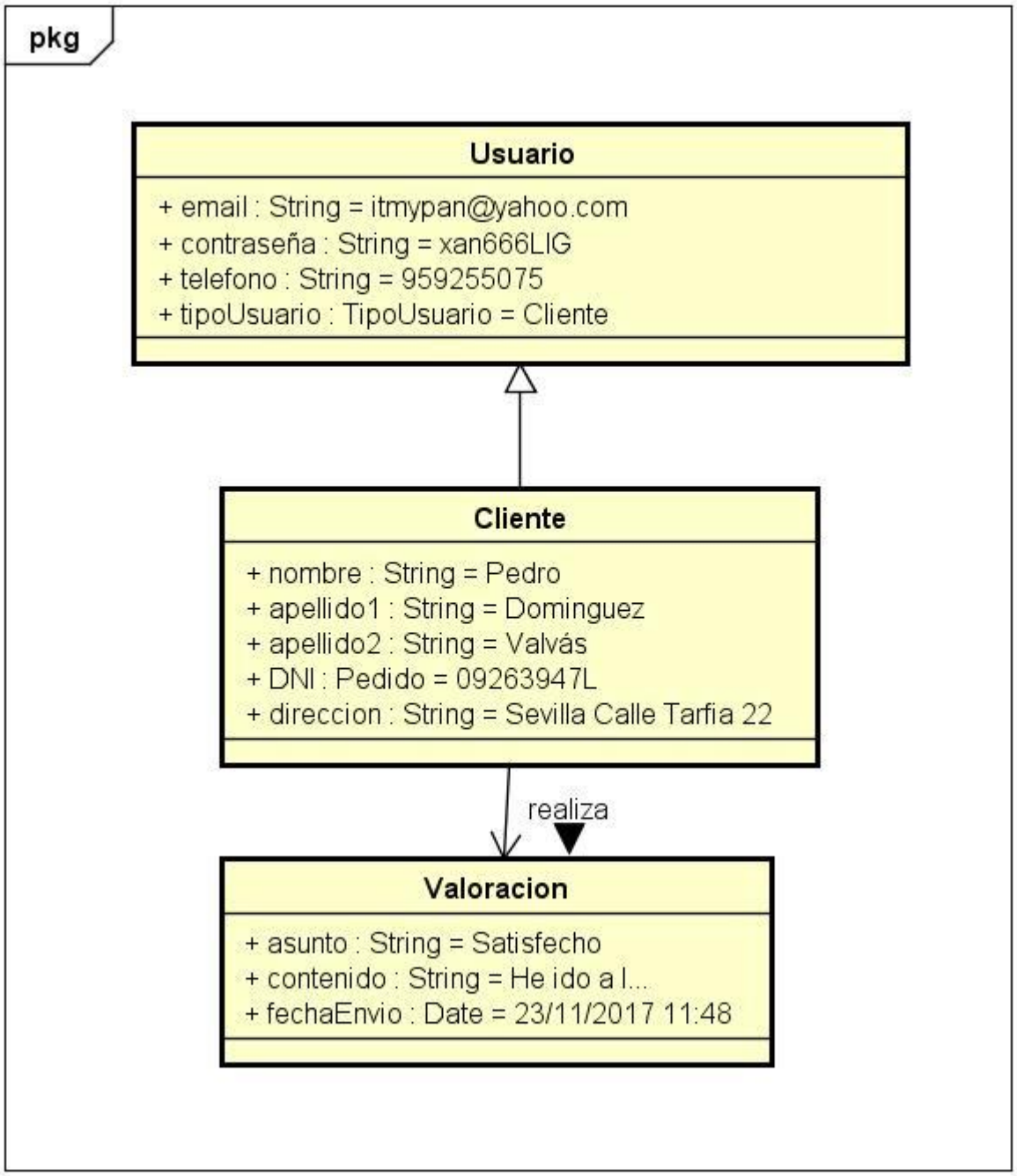
Escenario de Prueba 1

En este escenario el cliente Pedro Dominguez Valvás compra una moto “Triumph Boneville” y un par de guantes “Harley Davidson”.



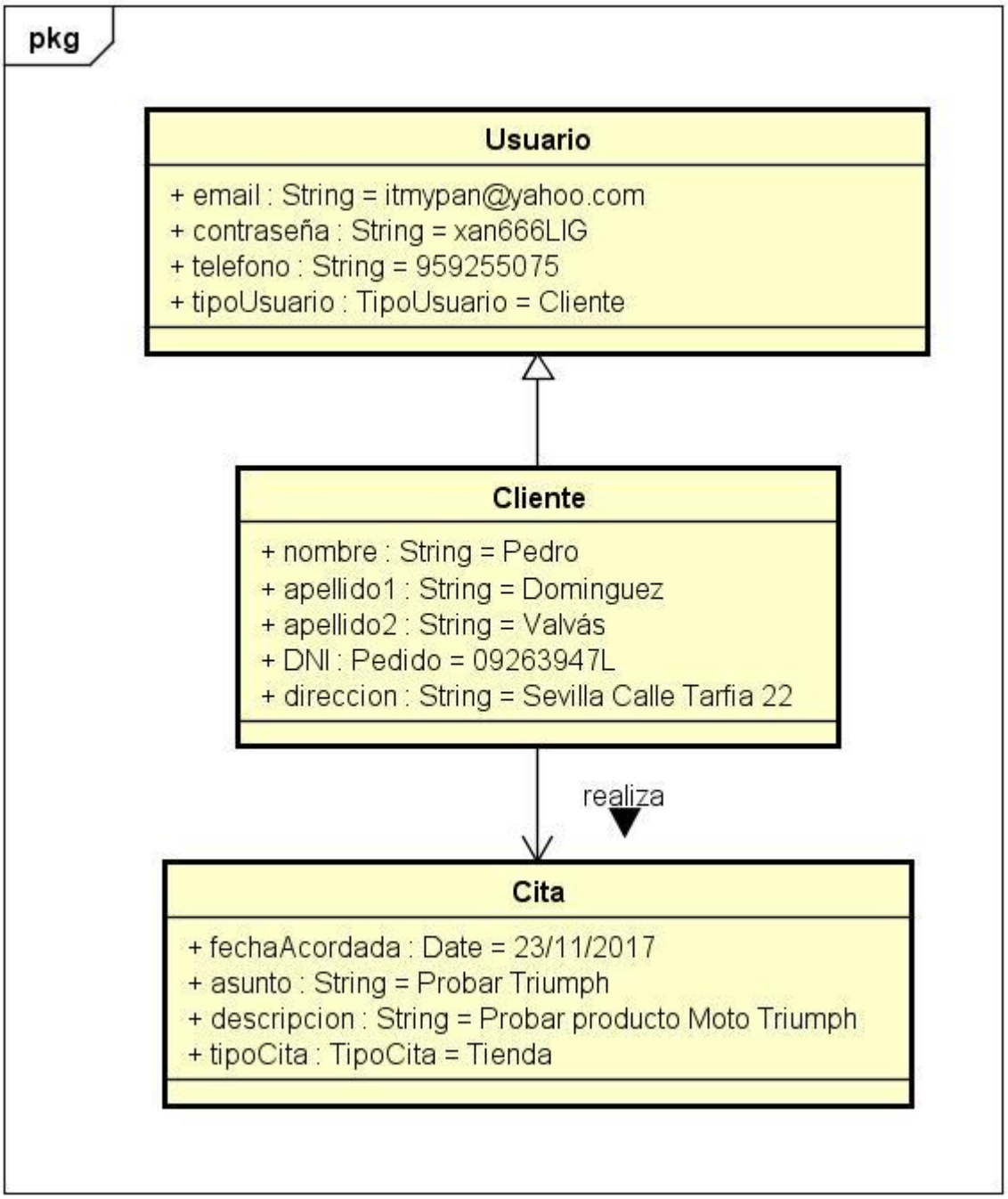
Escenario de Prueba 2

En este escenario el usuario Pedro Domínguez realiza una valoración al negocio sobre la compra que ha realizado en la tienda.



Escenario de Prueba 3

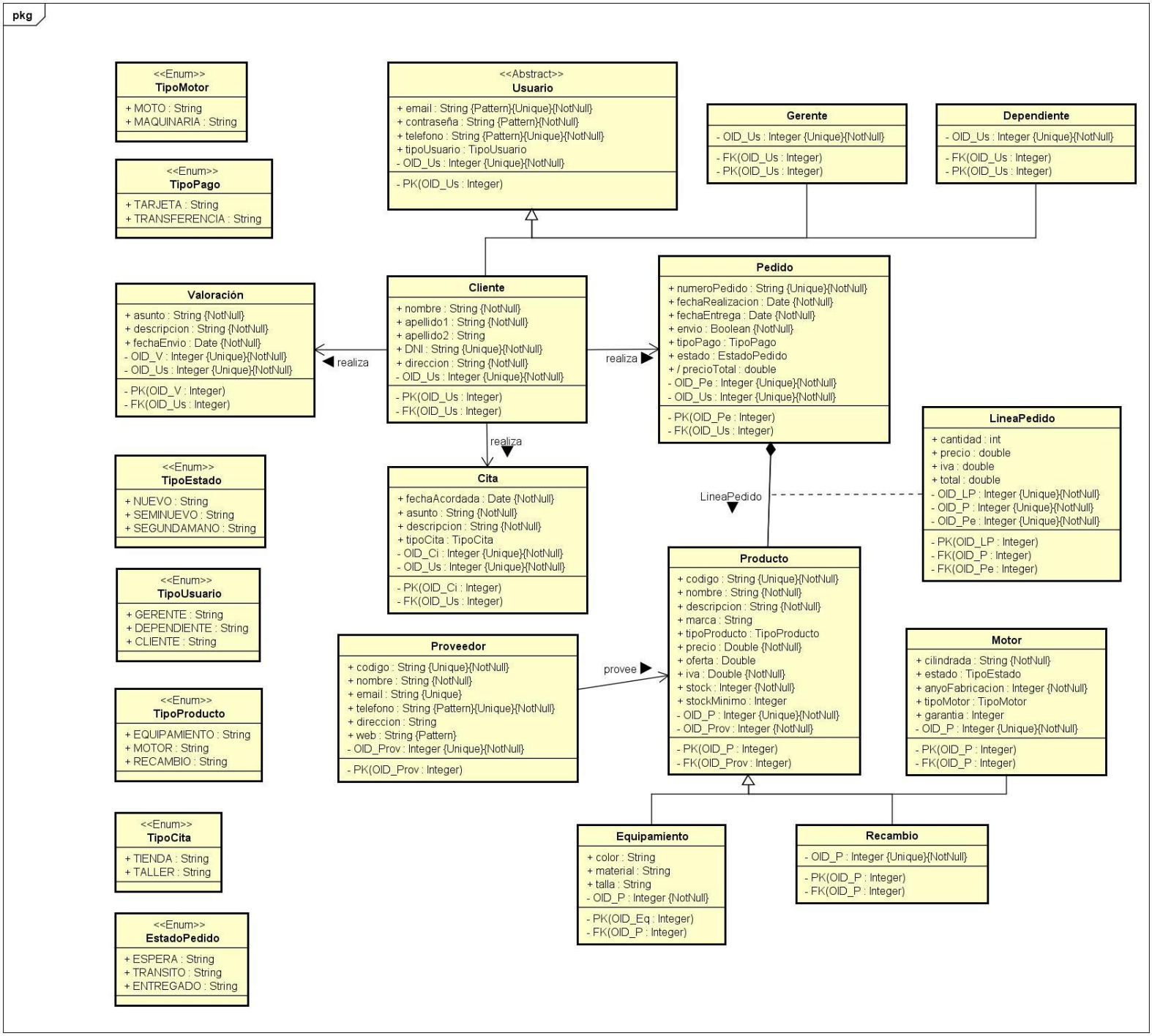
En este escenario de prueba el usuario Pedro Domínguez pide una cita para el día 23 de noviembre para ver la moto "Triumph Boneville".



7. MATRIZ DE TRAZABILIDAD

REQUISITOS	CLIENTE	GERENTE	)EPENDIENT	USUARIO	PEDIDOS	PRODUCTO	SUGERENCIA	CITA	C.USUARIO
RI-1		X		X					
RI-2			X		X				
RI-3		X				X			
RI-4	X	X						X	
RI-5		X		X					X
RI-6		X		X			X		
RN-1		X		X					
RN-2		X				X			
RN-3		X				X			
RN-4		X			X				
RN-5		X		X					
RN-6		X		X			X		

8. MODELO RELACIONAL EN 3FN



9. MODELO TECNOLÓGICO

1. Script de creación de tablas

```
DROP TABLE EQUIPAMIENTO  ;
DROP TABLE RECAMBIO  ;
DROP TABLE MOTOR  ;
DROP TABLE LINEAPEDIDO  ;
DROP TABLE PRODUCTO  ;
DROP TABLE PROVEEDOR  ;
DROP TABLE PEDIDO  ;
DROP TABLE CITA  ;
DROP TABLE VALORACION  ;
DROP TABLE CLIENTE  ;
DROP TABLE GERENTE  ;
DROP TABLE DEPENDIENTE  ;
DROP TABLE USUARIO  ;

CREATE TABLE USUARIO(
email VARCHAR2(50) NOT NULL UNIQUE,
contraseña VARCHAR(25),
telefono NUMBER(9) NOT NULL UNIQUE,
tipoUsuario VARCHAR2(25) CHECK( tipoUsuario
IN('Cliente','Gerente','Dependiente')),
OID_Us NUMBER(9) NOT NULL,
PRIMARY KEY(OID_Us),
constraint contraseña CHECK (REGEXP_LIKE(contraseña,'[[:alnum:]]')),
constraint contraseñaSoloNumeros CHECK
(REGEXP_LIKE(contraseña,'[[:alpha:]]')),
constraint contraseñaSoloLetras CHECK
(REGEXP_LIKE(contraseña,'[[:digit:]]'))
);

CREATE TABLE CLIENTE(
nombre VARCHAR2(25) NOT NULL,
primerApellido VARCHAR2(25) NOT NULL,
segundoApellido VARCHAR2(25) NOT NULL,
dni VARCHAR2(9) NOT NULL UNIQUE,
direccion VARCHAR2(25) NOT NULL,
OID_Us NUMBER(9) NOT NULL,
PRIMARY KEY(OID_Us),
FOREIGN KEY(OID_Us) REFERENCES USUARIO ON DELETE CASCADE,
CONSTRAINTS dni CHECK(REGEXP_LIKE(DNI, '^(\d{8}([A-Z]|[a-z]))$'))
);

CREATE TABLE DEPENDIENTE(
OID_Us NUMBER(9) NOT NULL,
PRIMARY KEY(OID_Us),
FOREIGN KEY(OID_Us) REFERENCES USUARIO ON DELETE CASCADE
```

```
);

CREATE TABLE GERENTE(
OID_Us NUMBER(9) NOT NULL,
PRIMARY KEY(OID_Us),
FOREIGN KEY(OID_Us) REFERENCES USUARIO ON DELETE CASCADE
);

CREATE TABLE VALORACION(
asunto VARCHAR(25) NOT NULL,
descripcion VARCHAR(500) NOT NULL,
fechaEnvio DATE DEFAULT SYSDATE,
OID_V NUMBER(9) NOT NULL,
OID_Us NUMBER(9) NOT NULL,
PRIMARY KEY(OID_V),
FOREIGN KEY(OID_Us) REFERENCES USUARIO ON DELETE CASCADE
);

CREATE TABLE CITA(
fechaAcordada DATE NOT NULL,
asunto VARCHAR(25) NOT NULL,
descripcion VARCHAR(500) NOT NULL,
tipoCita VARCHAR2(25) CHECK( tipoCita IN('Tienda', 'Taller')),
OID_Ci NUMBER(9) NOT NULL,
OID_Us NUMBER(9) NOT NULL,
PRIMARY KEY(OID_Ci),
FOREIGN KEY(OID_Us) REFERENCES USUARIO ON DELETE CASCADE
);

CREATE TABLE PROVEEDOR(
codigo VARCHAR(7) NOT NULL UNIQUE,
nombre VARCHAR(25) NOT NULL,
email VARCHAR(30) UNIQUE,
telefono VARCHAR2(9) NOT NULL UNIQUE,
direccion VARCHAR2(50),
web VARCHAR(50),
OID_Prov NUMBER(9) NOT NULL,
PRIMARY KEY(OID_Prov)
);

CREATE TABLE PEDIDO(
numeroPedido VARCHAR2(9) UNIQUE NOT NULL,
fechaRealizacion DATE DEFAULT SYSDATE,
fechaEntrega DATE NOT NULL,
envio VARCHAR2(25) CHECK( envio IN('True', 'False')),
tipoPago VARCHAR2(25) CHECK( tipoPago IN('Tarjeta', 'Transferencia')),
estado VARCHAR2(25) DEFAULT 'Espera' CHECK( estado IN('Espera',
'Transito', 'Entregado')),
precioTotal NUMBER(6,2) ,
OID_Pe NUMBER(9) NOT NULL,
OID_Us NUMBER(9) NOT NULL,
PRIMARY KEY(OID_Pe),
FOREIGN KEY(OID_Us) REFERENCES USUARIO ON DELETE CASCADE
```

```
);

CREATE TABLE PRODUCTO(
codigo VARCHAR2(9) NOT NULL UNIQUE,
nombre VARCHAR2(25) NOT NULL,
descripcion VARCHAR2(50) NOT NULL,
marca VARCHAR2(25),
tipoProducto VARCHAR2(25) CHECK( tipoProducto IN('Equipamiento', 'Motor',
'Recambio')),
precio NUMBER(*,2) NOT NULL,
oferta NUMBER(*,2),
iva NUMBER(*,2) NOT NULL,
stock NUMBER(2) NOT NULL,
stockMinimo NUMBER(2) NOT NULL,
OID_P NUMBER(9) NOT NULL,
OID_Prov NUMBER(9) NOT NULL,
PRIMARY KEY(OID_P),
FOREIGN KEY(OID_Prov) REFERENCES PROVEEDOR ON DELETE CASCADE,
CONSTRAINT stockNegativo CHECK(stock>0 or stock=0)
);

CREATE TABLE LINEAPEDIDO(
cantidad NUMBER(2,0) NOT NULL,
precio NUMBER(6,2) NOT NULL,
iva NUMBER(*,2) NOT NULL,
OID_LP NUMBER(9) NOT NULL,
OID_P NUMBER(9) NOT NULL,
OID_Pe NUMBER(9) NOT NULL,
PRIMARY KEY(OID_LP),
FOREIGN KEY(OID_P) REFERENCES PRODUCTO ON DELETE CASCADE,
FOREIGN KEY(OID_Pe) REFERENCES PEDIDO ON DELETE CASCADE
);

CREATE TABLE MOTOR(
cilindrada VARCHAR2(25) NOT NULL,
estado VARCHAR2(25) CHECK( estado IN('Nuevo', 'Seminuevo', 'Segunda
mano')),
anyoFabricacion NUMBER(4,0) NOT NULL,
tipoMotor VARCHAR2(25) CHECK( tipoMotor IN('Moto', 'Maquinaria')),
garantia NUMBER(1,0),
OID_P NUMBER(9) NOT NULL,
PRIMARY KEY(OID_P),
FOREIGN KEY(OID_P) REFERENCES PRODUCTO ON DELETE CASCADE,
CONSTRAINT maximoGarantia CHECK(garantia>0 OR garantia<7)
);

CREATE TABLE EQUIPAMIENTO(
color VARCHAR2(25) NOT NULL,
material VARCHAR2(25) NOT NULL,
talla VARCHAR2(25) NOT NULL,
OID_P NUMBER(9) NOT NULL,
PRIMARY KEY(OID_P),
FOREIGN KEY(OID_P) REFERENCES PRODUCTO ON DELETE CASCADE
```

```
);

CREATE TABLE RECAMBIO(
OID_P NUMBER(9) NOT NULL,
PRIMARY KEY(OID_P),
FOREIGN KEY(OID_P) REFERENCES PRODUCTO ON DELETE CASCADE
);
```

2. Script de creación secuencias y triggers asociados a la gestión de secuencias

```
DROP SEQUENCE sec_usuario;
DROP SEQUENCE sec_valoracion;
DROP SEQUENCE sec_cita;
DROP SEQUENCE sec_pedido;
DROP SEQUENCE sec_lineaPedido;
DROP SEQUENCE sec_producto;
DROP SEQUENCE sec_proveedor;
DROP SEQUENCE sec_equipamiento;

CREATE SEQUENCE sec_usuario INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_valoracion INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_cita INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_pedido INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_lineaPedido INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_producto INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_proveedor INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_equipamiento INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER crea_oid_usuario
BEFORE INSERT ON USUARIO
FOR EACH ROW
BEGIN
:NEW.OID_Us:=sec_usuario.NEXTVAL;
END;
/
CREATE OR REPLACE TRIGGER crea_oid_valoracion
BEFORE INSERT ON VALORACION
FOR EACH ROW
BEGIN
:NEW.OID_V:=sec_valoracion.NEXTVAL;
END;
/

CREATE OR REPLACE TRIGGER crea_oid_cita
BEFORE INSERT ON CITA
FOR EACH ROW
BEGIN
:NEW.OID_Ci:=sec_cita.NEXTVAL;
END;
/
```

```
CREATE OR REPLACE TRIGGER crea_oid_pedido
BEFORE INSERT ON PEDIDO
FOR EACH ROW
BEGIN
:NEW.OID_Pe:=sec_pedido.NEXTVAL;
END;
/

CREATE OR REPLACE TRIGGER crea_oid_lineaPedido
BEFORE INSERT ON LINEAPEDIDO
FOR EACH ROW
BEGIN
:NEW.OID_LP:=sec_lineaPedido.NEXTVAL;
END;
/

CREATE OR REPLACE TRIGGER crea_oid_producto
BEFORE INSERT ON PRODUCTO
FOR EACH ROW
BEGIN
:NEW.OID_P:=sec_producto.NEXTVAL;
END;
/

CREATE OR REPLACE TRIGGER crea_oid_proveedor
BEFORE INSERT ON PROVEEDOR
FOR EACH ROW
BEGIN
:NEW.OID_Prov:=sec_proveedor.NEXTVAL;
END;
/
```

3. Script de creación de funciones y procedimientos

```
--  RF-3. Registro en el sistema
--
--  Como usuario,
--  quiero registrarme en el sistema,
--  para facilitar el pedido en futuros pedidos
--
--

create or replace procedure crearCliente(
 xemail IN USUARIO.EMAIL%TYPE,
 xtelefono IN USUARIO.TELEFONO%TYPE,
 xcontraseña IN USUARIO."CONTRASEÑA"%TYPE,
 xtipoUsuario IN USUARIO.TIPOUSUARIO%TYPE,
 xnombre IN CLIENTE.NOMBRE%TYPE,
 xprimerApellido IN CLIENTE.PRIMERAPELLIDO%TYPE,
 xsegundoApellido IN CLIENTE.SEGUNDOAPELLIDO%TYPE,
 xdni IN CLIENTE.DNI%TYPE,
 xdireccion IN CLIENTE.DIRECCION%TYPE) IS
```

```
BEGIN
 INSERT INTO USUARIO (email,contraseña,telefono,tipoUsuario)
 VALUES (xemail,xcontraseña,xtelefono,xtipoUsuario);
 INSERT INTO CLIENTE
 (nombre,primerApellido,segundoApellido,dni,direccion,OID_US)
 VALUES (xnombre,xprimerApellido,xsegundoApellido,xdni,xdireccion,(SELECT
OID_Us FROM USUARIO WHERE email = xemail));
 COMMIT WORK;
End crearCliente;
/

create or replace procedure crearDependiente(
 xemail IN USUARIO.EMAIL%TYPE,
 xtelefono IN USUARIO.TELEFONO%TYPE,
 xcontraseña IN USUARIO."CONTRASEÑA"%TYPE
 ) IS
BEGIN
 INSERT INTO USUARIO (email,contraseña,telefono,tipoUsuario)
 VALUES (xemail,xcontraseña,xtelefono,'Dependiente');
 INSERT INTO DEPENDIENTE (OID_Us)
 VALUES ((SELECT OID_Us FROM USUARIO WHERE email = xemail));
 COMMIT WORK;
End crearDependiente;
/

create or replace procedure crearGerente(
 xemail IN USUARIO.EMAIL%TYPE,
 xtelefono IN USUARIO.TELEFONO%TYPE,
 xcontraseña IN USUARIO."CONTRASEÑA"%TYPE
 ) IS
BEGIN
 INSERT INTO USUARIO (email,contraseña,telefono,tipoUsuario) VALUES
(xemail,xcontraseña,xtelefono,'Gerente');
INSERT INTO GERENTE (OID_Us)
VALUES ((SELECT OID_Us FROM USUARIO WHERE email = xemail));
 COMMIT WORK;
End crearGerente;
/

create or replace procedure crearEquipamiento(
  xcodigo IN PRODUCTO.CODIGO%TYPE,
  xnombre IN PRODUCTO.NOMBRE%TYPE,
  xdescripcion IN PRODUCTO.DESCRIPCION%TYPE,
  xmarca IN PRODUCTO.MARCA%TYPE,
  xprecio IN PRODUCTO.PRECIO%TYPE,
  xoferta IN PRODUCTO.OFERTA%TYPE,
  xiva IN PRODUCTO.IVA%TYPE,
  xstock IN PRODUCTO.STOCK%TYPE,
  xstockMinimo IN PRODUCTO.STOCKMINIMO%TYPE,
  xOID_Prov IN PRODUCTO.OID_PROV%TYPE,
  xcolor IN EQUIPAMIENTO.COLOR%TYPE,
  xmaterial IN EQUIPAMIENTO.MATERIAL%TYPE,
  xtalla IN EQUIPAMIENTO.TALLA%TYPE
```

```
  ) IS
BEGIN
 INSERT INTO PRODUCTO
(codigo,nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stoc
kMinimo,OID_Prov)
 VALUES
(xcodigo,xnombre,xdescripcion,xmarca,'Equipamiento',xprecio,xoferta,xiva,x
stock,xstockMinimo,xOID_Prov);
INSERT INTO EQUIPAMIENTO (color,material,talla,OID_P)
VALUES (xcolor,xmaterial,xtalla,(SELECT OID_P FROM PRODUCTO WHERE
codigo=xcodigo));
 COMMIT WORK;
End crearEquipamiento;
/

create or replace procedure crearRecambio(
  xcodigo IN PRODUCTO.CODIGO%TYPE,
  xnombre IN PRODUCTO.NOMBRE%TYPE,
  xdescripcion IN PRODUCTO.DESCRIPCION%TYPE,
  xmarca IN PRODUCTO.MARCA%TYPE,
  xprecio IN PRODUCTO.PRECIO%TYPE,
  xoferta IN PRODUCTO.OFERTA%TYPE,
  xiva IN PRODUCTO.IVA%TYPE,
  xstock IN PRODUCTO.STOCK%TYPE,
  xstockMinimo IN PRODUCTO.STOCKMINIMO%TYPE,
  xOID_Prov IN PRODUCTO.OID_PROV%TYPE) IS
BEGIN
 INSERT INTO PRODUCTO
(codigo,nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stoc
kMinimo,OID_Prov) VALUES
(xcodigo,xnombre,xdescripcion,xmarca,'Recambio',xprecio,xoferta,xiva,xstoc
k,xstockMinimo,xOID_Prov);
INSERT INTO RECAMBIO (OID_P)
VALUES((SELECT OID_P FROM PRODUCTO WHERE codigo=xcodigo));
 COMMIT WORK;
End crearRecambio;
/

create or replace procedure crearMotor(
  xcodigo IN PRODUCTO.CODIGO%TYPE,
  xnombre IN PRODUCTO.NOMBRE%TYPE,
  xdescripcion IN PRODUCTO.DESCRIPCION%TYPE,
  xmarca IN PRODUCTO.MARCA%TYPE,
  xprecio IN PRODUCTO.PRECIO%TYPE,
  xoferta IN PRODUCTO.OFERTA%TYPE,
  xiva IN PRODUCTO.IVA%TYPE,
  xstock IN PRODUCTO.STOCK%TYPE,
  xstockMinimo IN PRODUCTO.STOCKMINIMO%TYPE,
  xOID_Prov IN PRODUCTO.OID_PROV%TYPE,
  xcilindrada IN MOTOR.CILINDRADA%TYPE,
  xestado IN MOTOR.ESTADO%TYPE,
  xanyoFabricacion IN MOTOR.ANYOFABRICACION%TYPE,
  xtipoMotor IN MOTOR.TIPOMOTOR%TYPE,
```

```
  xgarantia IN MOTOR.GARANTIA%TYPE
  ) IS
BEGIN
 INSERT INTO
PRODUCTO(codigo,nombre,descripcion,marca,tipoProducto,precio,oferta,iva,st
ock,stockMinimo,OID_Prov)
 VALUES
(xcodigo,xnombre,xdescripcion,xmarca,'Motor',xprecio,xoferta,xiva,xstock,x
stockMinimo,xOID_Prov);
INSERT INTO MOTOR
(cilindrada,estado,anyoFabricacion,tipoMotor,garantia,OID_P)
VALUES(xcilindrada,xestado,xanyoFabricacion,xtipoMotor,xgarantia,(SELECT
OID_P FROM PRODUCTO WHERE codigo=xcodigo));
 COMMIT WORK;
End crearMotor;
/
--RF-10. Realizar valoraciones
--
--Como cliente,
--quiero poder realizar una valoración,
--para así poder expresar mi opinión sobre el negocio.

create or replace procedure crearValoracion(

 xasunto IN VALORACION.ASUNTO%TYPE,
 xdescripcion IN VALORACION.DESCRIPCION%TYPE,
 xOID_Us IN VALORACION.OID_US%TYPE
 ) IS
BEGIN
 INSERT INTO VALORACION (asunto,descripcion,OID_Us)
 VALUES(xasunto,xdescripcion,xOID_Us);
 COMMIT WORK;
End crearValoracion;
/

--RF-11. Solicitar citas
--
--Como cliente,
--quiero poder solicitar una cita,
--para así poder contactar de forma presencial con la empresa.
--

create or replace procedure crearCita(

 xfechaAcordada IN CITA.FECHAACORDADA%TYPE,
 xasunto IN CITA.ASUNTO%TYPE,
 xdescripcion IN CITA.DESCRIPCION%TYPE,
 xtipoCita IN CITA.TIPOCITA%TYPE,
 xOID_Us IN CITA.OID_US%TYPE
 ) IS
BEGIN
 INSERT INTO CITA (fechaAcordada,asunto,descripcion,tipoCita,OID_US)
VALUES (xfechaAcordada,xasunto,xdescripcion,xtipoCita,xOID_US);
```

```
 COMMIT WORK;
End crearCita;
/
create or replace procedure crearPedido(
  xnumeroPedido IN PEDIDO.NUMEROPEDIDO%TYPE,
  xfechaEntrega  IN PEDIDO.FECHAENTREGA%TYPE,
  xenvio IN PEDIDO.ENVIO%TYPE,
  xtipoPago  IN PEDIDO.TIPOPAGO%TYPE,
  xOID_Us IN PEDIDO.OID_US%TYPE) IS
  BEGIN
  INSERT INTO PEDIDO
(numeroPedido,fechaEntrega,envio,tipoPago,estado,OID_Us)
  VALUES (xnumeroPedido,xfechaEntrega,xenvio,xtipoPago,'Espera',xOID_Us);
  COMMIT WORK;
  END crearPedido;
  /
create or replace procedure crearLineaPedido(
  xcantidad  IN LINEAPEDIDO.CANTIDAD%TYPE,
  xOID_P IN LINEAPEDIDO.OID_P%TYPE,
  xOID_Pe IN LINEAPEDIDO.OID_PE%TYPE) IS
  v_OID_LP LINEAPEDIDO.OID_LP%TYPE;
  BEGIN
  BEGIN
SELECT OID_LP INTO v_OID_LP FROM LINEAPEDIDO WHERE OID_P=xOID_P AND
OID_Pe=xOID_Pe;
EXCEPTION
WHEN NO_DATA_FOUND THEN
v_OID_LP:=0;
END;

IF v_OID_LP=0 THEN
  INSERT INTO LINEAPEDIDO(cantidad,precio,iva,OID_P,OID_Pe)
  VALUES (xcantidad,(SELECT PRECIO FROM PRODUCTO WHERE
OID_P=xOID_P),(SELECT IVA FROM PRODUCTO WHERE
OID_P=xOID_P),xOID_P,xOID_Pe);
ELSE
  UPDATE LineaPedido SET cantidad = cantidad+xcantidad WHERE
OID_LP=v_OID_LP;
  END IF;
  COMMIT WORK;
  END crearLineaPedido;
  /

create or replace procedure crearProveedor(
 xcodigo IN PROVEEDOR.CODIGO%TYPE,
 xnombre IN PROVEEDOR.NOMBRE%TYPE,
 xemail IN PROVEEDOR.EMAIL%TYPE,
 xtelefono IN PROVEEDOR.TELEFONO%TYPE,
 xdireccion IN PROVEEDOR.DIRECCION%TYPE,
 xweb IN PROVEEDOR.WEB%TYPE
 ) IS
BEGIN
 INSERT INTO PROVEEDOR (codigo,nombre,email,telefono,direccion,web)
```

```
 VALUES(xcodigo,xnombre,xemail,xtelefono,xdireccion,xweb);
 COMMIT WORK;
End crearProveedor;
/

--  RF-01
--  Como dependiente,
--  quiero gestionar pedido de un cliente en cada momento,
--  para tener un seguimiento exhaustivo del mismo.

create or replace procedure verPedidos IS
CURSOR C IS
SELECT
numeroPedido,fechaRealizacion,fechaEntrega,envio,tipoPago,estado,precioTot
al,OID_Pe FROM PEDIDO;
v_Cursor C%ROWTYPE;
BEGIN
OPEN C;
FETCH C INTO v_Cursor;
 DBMS_OUTPUT.PUT_LINE(RPAD('Numero del pedido:', 25) || RPAD('Fecha de
realizacion:', 25) || RPAD('Fecha de entrega', 25) ||
 RPAD('Envio:', 25) || RPAD('Tipo de pago:', 25) || RPAD('Estado:', 25) ||
RPAD('Precio Total:', 25) || RPAD('OID_Pe:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 135, '-'));
 WHILE C%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_Cursor.numeroPedido, 25) ||
RPAD(v_Cursor.fechaRealizacion, 25) || RPAD(v_Cursor.fechaEntrega, 25)
||RPAD(v_Cursor.envio, 25) ||
 RPAD(v_Cursor.tipoPago, 25) || RPAD(v_Cursor.estado, 25) ||
RPAD(v_Cursor.precioTotal, 25) || RPAD(v_Cursor.OID_Pe, 25));
 FETCH C INTO v_Cursor;
 END LOOP;
 CLOSE C;
END verPedidos;
/

--RF-4. Carrito de la compra
--
--Como usuario,
--quiero disponer de la opción de un carrito de la compra,
--para poder organizar un pedido.
create or replace procedure devolverCarrito(
v_OID_Pe IN PEDIDO.OID_Pe%TYPE)IS
v_PrecioTotal NUMBER(6,2);
 CURSOR C IS
 SELECT nombre, precio, cantidad, precioTotal FROM PEDIDO NATURAL JOIN
LineaPedido NATURAL JOIN Producto WHERE v_OID_Pe = OID_Pe;
 v_Carrito C%ROWTYPE;
BEGIN
 SELECT precioTotal INTO v_PrecioTotal FROM PEDIDO WHERE v_OID_Pe =
OID_Pe;
 OPEN C;
 FETCH C INTO v_Carrito;
```

```
 DBMS_OUTPUT.PUT_LINE(RPAD('Producto:', 25) || RPAD('Precio:', 25) ||
RPAD('Cantidad', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 135, '-'));
 WHILE C%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_Carrito.nombre, 25) || RPAD(v_Carrito.precio,
25) || RPAD(v_Carrito.cantidad, 25));
 FETCH C INTO v_Carrito;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE(RPAD('Precio total:',25) || RPAD(v_PrecioTotal,25));
 CLOSE C;
END devolverCarrito;
/

--RF-5. Añadir/eliminar/editar productos
--
--Como gerente,
--quiero poder modificar el catálogo de la tienda online dependiendo de
las necesidades de la tienda,
--para ofrecer productos a la venta.

create or replace procedure updateProducto(
  xOID_P IN PRODUCTO.OID_P%TYPE,
  xnombre IN PRODUCTO.NOMBRE%TYPE,
  xdescripcion IN PRODUCTO.DESCRIPCION%TYPE,
  xmarca IN PRODUCTO.MARCA%TYPE,
  xtipoProducto IN PRODUCTO.TIPOPRODUCTO%TYPE,
  xprecio IN PRODUCTO.PRECIO%TYPE,
  xoferta IN PRODUCTO.OFERTA%TYPE,
  xiva IN PRODUCTO.IVA%TYPE,
  xstock IN PRODUCTO.STOCK%TYPE,
  xstockMinimo IN PRODUCTO.STOCKMINIMO%TYPE,
  xOID_Prov IN PRODUCTO.OID_PROV%TYPE,
  xcilindrada IN MOTOR.CILINDRADA%TYPE,
  xestado IN MOTOR.ESTADO%TYPE,
  xanyoFabricacion IN MOTOR.ANYOFABRICACION%TYPE,
  xtipoMotor IN MOTOR.TIPOMOTOR%TYPE,
  xgarantia IN MOTOR.GARANTIA%TYPE,
  xcolor IN EQUIPAMIENTO.COLOR%TYPE,
  xmaterial IN EQUIPAMIENTO.MATERIAL%TYPE,
  xtalla IN EQUIPAMIENTO.TALLA%TYPE
  ) IS
BEGIN

UPDATE PRODUCTO set
nombre=xnombre,descripcion=xdescripcion,marca=xmarca,tipoProducto=xtipoPro
ducto,precio=xprecio,oferta=xoferta,iva=xiva,stock=xstock,
stockMinimo=xstockMinimo,OID_Prov=xOID_Prov
WHERE OID_P = xOID_P;
IF xtipoProducto='Equipamiento' THEN
UPDATE EQUIPAMIENTO set color=xcolor,material=xmaterial,talla=xtalla WHERE
OID_P = xOID_P;
ELSIF xtipoProducto='Motor' THEN
```

```
UPDATE MOTOR set
cilindrada=xcilindrada,estado=xestado,anyoFabricacion=xanyoFabricacion,tip
oMotor=xtipoMotor,garantia=xgarantia WHERE OID_P =xOID_P;
ELSIF xtipoProducto!='Recambio' THEN
 DBMS_OUTPUT.PUT_LINE(RPAD('Tipo producto incorrecto',50));
END IF;
 COMMIT WORK;
End updateProducto;
/

create or replace procedure borrarProducto(
xOID_P IN PRODUCTO.OID_P%TYPE
) IS
BEGIN
DELETE FROM Producto WHERE OID_P = xOID_P;
--   ON DELETE CASCADE NOS PERMITE QUE AL BORRAR EL PRODUCTO TAMBIEN SE
BORREN SUS CLASES HIJAS
End borrarProducto;
/


--RF-6. Listar productos
--
--Como cliente,
--quiero que el sistema me ofrezca un conjunto de productos,
--para poder seleccionarlos y comprarlos si lo deseo.


create or replace procedure listarProductos IS
CURSOR C IS
 SELECT
nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stockMinimo,
OID_Prov,color,material,talla
 FROM EQUIPAMIENTO NATURAL JOIN PRODUCTO;
 v_equipo C%ROWTYPE;
 CURSOR D IS
 SELECT
nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stockMinimo,
OID_Prov,cilindrada,estado,anyoFabricacion,tipoMotor,garantia
 FROM MOTOR NATURAL JOIN PRODUCTO;
 v_motor D%ROWTYPE;
  CURSOR E IS
 SELECT
nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stockMinimo,
OID_Prov
 FROM  RECAMBIO NATURAL JOIN PRODUCTO ;
 v_recambio E%ROWTYPE;

BEGIN
 OPEN C;
 FETCH C INTO v_equipo;
 DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Descripcion:', 25) ||
RPAD('Marca:', 25) || RPAD('Precio:', 25) || RPAD('Oferta:', 25)||
```

```
 RPAD('Color:', 25)|| RPAD('Material:', 25)|| RPAD('Talla:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
 WHILE C%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_equipo.nombre, 25) ||
RPAD(v_equipo.descripcion, 25) || RPAD(v_equipo.marca, 25) ||
RPAD(v_equipo.precio,25) || RPAD(v_equipo.oferta, 25)
|| RPAD(v_equipo.color, 25) || RPAD(v_equipo.material, 25) ||
RPAD(v_equipo.talla, 25));
 FETCH C INTO v_equipo;
 END LOOP;
 CLOSE C;
  OPEN D;
 FETCH D INTO v_motor;
 DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Descripcion:', 25) ||
RPAD('Marca:', 25) || RPAD('Precio:', 25) || RPAD('Oferta:', 25)||
 RPAD('Cilindradas:', 25)|| RPAD('Estado:', 25)|| RPAD('Año de
fabricacion:', 25)||  RPAD('Garantia:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
 WHILE D%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_motor.nombre, 25) ||
RPAD(v_motor.descripcion, 25) || RPAD(v_motor.marca, 25) ||
RPAD(v_motor.precio,25) || RPAD(v_motor.oferta, 25)
|| RPAD(v_motor.cilindrada, 25) || RPAD(v_motor.estado, 25) ||
RPAD(v_motor.anyoFabricacion, 25)|| RPAD(v_motor.garantia, 25));
 FETCH D INTO v_motor;
 END LOOP;
 CLOSE D;
   OPEN E;
 FETCH E INTO v_recambio;
 DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Descripcion:', 25) ||
RPAD('Marca:', 25) || RPAD('Precio:', 25) || RPAD('Oferta:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
 WHILE E%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_recambio.nombre, 25) ||
RPAD(v_recambio.descripcion, 25) || RPAD(v_recambio.marca, 25) ||
RPAD(v_recambio.precio,25) || RPAD(v_recambio.oferta, 25));
 FETCH E INTO v_recambio;
 END LOOP;
 CLOSE E;

END listarProductos;
/

--RF-7. Realizar pedido
--
--Como usuario,
--quiero poder realizar el pedido,
--para adquirir producto que desee.


create or replace procedure anadirCarrito(
v_OID_Us IN USUARIO.OID_Us%TYPE,
v_OID_P IN PRODUCTO.OID_P%TYPE,
```

```
  xnumeroPedido IN PEDIDO.NUMEROPEDIDO%TYPE,
  xfechaEntrega  IN PEDIDO.FECHAENTREGA%TYPE,
  xenvio IN PEDIDO.ENVIO%TYPE,
  xtipoPago  IN PEDIDO.TIPOPAGO%TYPE,
  xOID_Us IN PEDIDO.OID_US%TYPE)IS
v_OID_Pe PEDIDO.OID_Pe%TYPE:=NULL;
ccc PEDIDO.OID_Us%TYPE;
BEGIN
BEGIN
SELECT OID_Pe INTO v_OID_Pe FROM PEDIDO WHERE OID_Us=v_OID_Us;
EXCEPTION
WHEN NO_DATA_FOUND THEN
v_OID_Pe:=NULL;
END;

IF v_OID_Pe=NULL THEN
 crearPedido(xnumeroPedido,xfechaEntrega,xenvio,xtipoPago,v_OID_Us);
 END IF;
 SELECT OID_Pe INTO ccc FROM PEDIDO WHERE OID_Us = v_OID_Us;
 CREARLINEAPEDIDO(1,v_OID_P,ccc);

END anadirCarrito;
/

--RF-8. Gestionar citas
--
--Como gerente,
--quiero poder gestionar las citas con los clientes,
--para poder adecuarlas a mi horario.

create or replace procedure gestionarCitas IS
auxnom VARCHAR2(25);
auxnom2 VARCHAR2(25);
 CURSOR C IS
 SELECT fechaAcordada,asunto,descripcion,tipoCita,OID_US FROM CITA;
 v_Cita C%ROWTYPE;
BEGIN
 OPEN C;
 FETCH C INTO v_Cita;
 DBMS_OUTPUT.PUT_LINE(RPAD('Asunto:', 25) || RPAD('Cliente:', 50) ||
RPAD('Tipo', 25)|| RPAD('Fecha:', 25)|| RPAD('Descripcion:', 50));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 135, '-'));
 WHILE C%FOUND LOOP
 Select nombre,PRIMERAPELLIDO INTO auxnom,auxnom2 FROM CLIENTE WHERE
OID_US=v_Cita.OID_US;
 DBMS_OUTPUT.PUT_LINE(RPAD(v_Cita.asunto, 25) || RPAD(auxnom, 25) ||
RPAD(auxnom2, 25) || RPAD(v_Cita.tipoCita, 25)||
RPAD(v_Cita.fechaAcordada, 25)|| RPAD(v_Cita.descripcion, 50));
 FETCH C INTO v_Cita;
 END LOOP;
 CLOSE C;
END gestionarCitas;
/
```

```
--RF-9: Visualizar valoraciones
--
--Como gerente,
--quiero poder visualizar las valoraciones de los clientes,
--para poder mejorar mi negocio.

create or replace procedure visualizarValoraciones IS
auxnom VARCHAR2(25);
auxnom2 VARCHAR2(25);
 CURSOR C IS
 SELECT fechaEnvio,asunto,descripcion,OID_US FROM Valoracion;
 v_Valoracion C%ROWTYPE;
BEGIN
 OPEN C;
 FETCH C INTO v_Valoracion;
 DBMS_OUTPUT.PUT_LINE(RPAD('Asunto:', 25) || RPAD('Cliente:', 50) ||
RPAD('Fecha:', 25)|| RPAD('Descripcion:', 50));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 135, '-'));
 WHILE C%FOUND LOOP
 Select nombre,PRIMERAPELLIDO INTO auxnom,auxnom2 FROM CLIENTE WHERE
OID_US=v_Valoracion.OID_US;
 DBMS_OUTPUT.PUT_LINE(RPAD(v_Valoracion.asunto, 25) || RPAD(auxnom, 25) ||
RPAD(auxnom2, 25) || RPAD(v_Valoracion.fechaEnvio, 25)||
RPAD(v_Valoracion.descripcion, 50));
 FETCH C INTO v_Valoracion;
 END LOOP;
 CLOSE C;
END visualizarValoraciones;
/

--RF-12. Consultar pedidos
--
--Como usuario,
--quiero poder ver el historial de pedidos que he efectuado,
--para poder consultarlos si lo deseo.

create or replace procedure verPedido(
v_OID_Us IN USUARIO.OID_US%TYPE) IS
CURSOR C IS
SELECT
numeroPedido,fechaRealizacion,fechaEntrega,envio,tipoPago,estado,precioTot
al,OID_Pe FROM PEDIDO WHERE OID_Us = v_OID_Us;
v_Cursor C%ROWTYPE;
BEGIN
OPEN C;
FETCH C INTO v_Cursor;
 DBMS_OUTPUT.PUT_LINE(RPAD('Numero del pedido:', 25) || RPAD('Fecha de
realizacion:', 25) || RPAD('Fecha de entrega', 25) ||
 RPAD('Envio:', 25) || RPAD('Tipo de pago:', 25) || RPAD('Estado:', 25) ||
RPAD('Precio Total:', 25) || RPAD('OID_Pe:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 135, '-'));
 WHILE C%FOUND LOOP
```

```
 DBMS_OUTPUT.PUT_LINE(RPAD(v_Cursor.numeroPedido, 25) ||
RPAD(v_Cursor.fechaRealizacion, 25) || RPAD(v_Cursor.fechaEntrega, 25)
||RPAD(v_Cursor.envio, 25) ||
 RPAD(v_Cursor.tipoPago, 25) || RPAD(v_Cursor.estado, 25) ||
RPAD(v_Cursor.precioTotal, 25) || RPAD(v_Cursor.OID_Pe, 25));
 FETCH C INTO v_Cursor;
 END LOOP;
 CLOSE C;
END verPedido;
/

--RF-13: Consultar ofertas
--
--Como usuario,
--quiero poder ver los productos que estén en oferta,
--para adquirirlos si lo deseo.

create or replace procedure verOfertas IS
CURSOR C IS
 SELECT
nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stockMinimo,
OID_Prov,color,material,talla
 FROM EQUIPAMIENTO NATURAL JOIN PRODUCTO WHERE oferta !=null;
 v_equipo C%ROWTYPE;
 CURSOR D IS
 SELECT
nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stockMinimo,
OID_Prov,cilindrada,estado,anyoFabricacion,tipoMotor,garantia
 FROM MOTOR NATURAL JOIN PRODUCTO WHERE oferta !=null;
 v_motor D%ROWTYPE;
  CURSOR E IS
 SELECT
nombre,descripcion,marca,tipoProducto,precio,oferta,iva,stock,stockMinimo,
OID_Prov
 FROM  RECAMBIO NATURAL JOIN PRODUCTO  WHERE oferta !=null;
 v_recambio E%ROWTYPE;

BEGIN
 OPEN C;
 FETCH C INTO v_equipo;
 DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Descripcion:', 25) ||
RPAD('Marca:', 25) || RPAD('Precio:', 25) || RPAD('Oferta:', 25)||
 RPAD('Color:', 25)|| RPAD('Material:', 25)|| RPAD('Talla:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
 WHILE C%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_equipo.nombre, 25) ||
RPAD(v_equipo.descripcion, 25) || RPAD(v_equipo.marca, 25) ||
RPAD(v_equipo.precio,25) || RPAD(v_equipo.oferta, 25)
|| RPAD(v_equipo.color, 25) || RPAD(v_equipo.material, 25) ||
RPAD(v_equipo.talla, 25));
 FETCH C INTO v_equipo;
 END LOOP;
 CLOSE C;
```

```
  OPEN D;
 FETCH D INTO v_motor;
 DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Descripcion:', 25) ||
RPAD('Marca:', 25) || RPAD('Precio:', 25) || RPAD('Oferta:', 25)||
 RPAD('Cilindradas:', 25)|| RPAD('Estado:', 25)|| RPAD('Año de
fabricacion:', 25)||  RPAD('Garantia:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
 WHILE D%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_motor.nombre, 25) ||
RPAD(v_motor.descripcion, 25) || RPAD(v_motor.marca, 25) ||
RPAD(v_motor.precio,25) || RPAD(v_motor.oferta, 25)
|| RPAD(v_motor.cilindrada, 25) || RPAD(v_motor.estado, 25) ||
RPAD(v_motor.anyoFabricacion, 25)|| RPAD(v_motor.garantia, 25));
 FETCH D INTO v_motor;
 END LOOP;
 CLOSE D;
   OPEN E;
 FETCH E INTO v_recambio;
 DBMS_OUTPUT.PUT_LINE(RPAD('Nombre:', 25) || RPAD('Descripcion:', 25) ||
RPAD('Marca:', 25) || RPAD('Precio:', 25) || RPAD('Oferta:', 25));
 DBMS_OUTPUT.PUT_LINE(LPAD('-', 120, '-'));
 WHILE E%FOUND LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(v_recambio.nombre, 25) ||
RPAD(v_recambio.descripcion, 25) || RPAD(v_recambio.marca, 25) ||
RPAD(v_recambio.precio,25) || RPAD(v_recambio.oferta, 25));
 FETCH E INTO v_recambio;
 END LOOP;
 CLOSE E;

END verOfertas;
/
```

4. Script de creación de triggers no asociados a secuencias

```
CREATE OR REPLACE TRIGGER actualizacionStock
  AFTER INSERT OR DELETE OR UPDATE ON LINEAPEDIDO
  FOR EACH ROW
  DECLARE
  contador integer;
  BEGIN
  IF (inserting OR updating)
    THEN
    SELECT stock INTO contador FROM PRODUCTO WHERE OID_P=:NEW.OID_P;
    IF(contador<:NEW.cantidad)
      THEN
      RAISE_APPLICATION_ERROR(-20004,'No se puede añadir más cantidad de
la existente');
    END IF;

    UPDATE PRODUCTO SET stock= stock - :new.cantidad WHERE
    OID_P=:NEW.OID_P;
```

```
  END IF;
  IF (deleting OR updating)
    THEN
    UPDATE PRODUCTO SET stock= stock + :OLD.cantidad WHERE
    OID_P=:OLD.OID_P;
  END IF;
END;
/



CREATE OR REPLACE TRIGGER precioTotalPedido
  AFTER INSERT OR DELETE OR UPDATE ON LINEAPEDIDO
  FOR EACH ROW
  DECLARE
  columnaProducto PRODUCTO%ROWTYPE;
  BEGIN
  IF (inserting or updating)
    then
    select *into columnaProducto from PRODUCTO where
    OID_P=:new.OID_P;
    update PEDIDO set precioTotal=precioTotal+
    (:new.cantidad*columnaProducto.precio) where OID_PE=:new.OID_PE;
  END IF;
  IF (updating )
    then
    select *into columnaProducto from PRODUCTO where
    OID_P=:new.OID_P;
    update PEDIDO set precioTotal=precioTotal-
    (:old.cantidad*columnaProducto.precio) where OID_PE=:old.OID_PE;
  END IF;
  IF(deleting)
    then
    select *into columnaProducto from PRODUCTO where
    OID_P=:old.OID_P;
    update PEDIDO set precioTotal=precioTotal-
    (:old.cantidad*columnaProducto.precio) where OID_PE=:old.OID_PE;
  END IF;
END;
/
```

5. Script de paquetes de pruebas

```
CREATE OR REPLACE FUNCTION ASSERT_EQUALS (v_Salida BOOLEAN, salidaEsperada
BOOLEAN)
RETURN VARCHAR2
AS
BEGIN
  IF v_Salida = salidaEsperada THEN
    RETURN 'ÉXITO';
  ELSE
    RETURN 'FALLO';
  END IF;
END;
/

-- PRUEBAS CLIENTE-----------------------------------------------
------------------------------------------------------------------

CREATE OR REPLACE PACKAGE PCK_Cliente
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_email IN USUARIO.EMAIL%TYPE,
v_contraseña IN USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, v_nombre IN CLIENTE.NOMBRE%TYPE,
  v_primerApellido IN CLIENTE.PRIMERAPELLIDO%TYPE, v_segundoApellido IN
CLIENTE.SEGUNDOAPELLIDO%TYPE,
  v_dni IN CLIENTE.DNI%TYPE, v_direccion IN CLIENTE.DIRECCION%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, v_email IN USUARIO.EMAIL%TYPE, v_contraseña IN
USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, v_nombre IN CLIENTE.NOMBRE%TYPE,
  v_primerApellido IN CLIENTE.PRIMERAPELLIDO%TYPE, v_segundoApellido IN
CLIENTE.SEGUNDOAPELLIDO%TYPE,
  v_dni IN CLIENTE.DNI%TYPE, v_direccion IN CLIENTE.DIRECCION%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, salidaEsperada BOOLEAN);
END;
/

CREATE OR REPLACE PACKAGE BODY PCK_Cliente
AS
  v_Salida BOOLEAN := TRUE;
  CURSOR C IS
  SELECT * FROM USUARIO NATURAL JOIN CLIENTE;
  v_Usuario C%ROWTYPE;
  PROCEDURE Inicializar AS
  BEGIN
  DELETE FROM USUARIO WHERE tipoUsuario='Cliente';
```

```
    END Inicializar;

  PROCEDURE Insertar (nombrePrueba VARCHAR2, v_email IN
USUARIO.EMAIL%TYPE, v_contraseña IN USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, v_nombre IN CLIENTE.NOMBRE%TYPE,
  v_primerApellido IN CLIENTE.PRIMERAPELLIDO%TYPE, v_segundoApellido IN
CLIENTE.SEGUNDOAPELLIDO%TYPE,
  v_dni IN CLIENTE.DNI%TYPE, v_direccion IN CLIENTE.DIRECCION%TYPE,
salidaEsperada BOOLEAN)
  AS
  BEGIN

crearCliente(v_email,v_telefono,v_contraseña,'Cliente',v_nombre,v_primerAp
ellido,v_segundoApellido,
    v_dni,v_direccion);
    SELECT * INTO v_Usuario FROM Usuario NATURAL JOIN CLIENTE WHERE
v_email=Usuario.EMAIL;
    IF (v_Usuario.email <> v_email
    OR v_Usuario.contraseña <> v_contraseña
    OR v_Usuario.telefono <> v_telefono
    OR v_Usuario.nombre <> v_nombre
    OR v_Usuario.primerApellido <> v_primerApellido
    OR v_Usuario.segundoApellido <> v_segundoApellido
    OR v_Usuario.dni <> v_dni
    OR v_Usuario.direccion <> v_direccion)
    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT WORK;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

  PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, v_email IN USUARIO.EMAIL%TYPE, v_contraseña IN
USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, v_nombre IN CLIENTE.NOMBRE%TYPE,
  v_primerApellido IN CLIENTE.PRIMERAPELLIDO%TYPE, v_segundoApellido IN
CLIENTE.SEGUNDOAPELLIDO%TYPE,
  v_dni IN CLIENTE.DNI%TYPE, v_direccion IN CLIENTE.DIRECCION%TYPE,
salidaEsperada BOOLEAN) AS
  BEGIN
    UPDATE Usuario set EMAIL = v_email, "CONTRASEÑA"=v_contraseña,
TELEFONO=v_telefono
      WHERE OID_Us = v_OID_Us;
    UPDATE Cliente set NOMBRE=v_nombre, PRIMERAPELLIDO=v_primerApellido,
SEGUNDOAPELLIDO=v_segundoApellido, DNI= v_dni,
      DIRECCION= v_direccion WHERE OID_Us = v_OID_Us;
```

```
    SELECT * INTO v_Usuario FROM Usuario NATURAL JOIN CLIENTE WHERE
v_OID_Us=OID_Us;
    IF (v_Usuario.email <> v_email
    OR v_Usuario.contraseña <> v_contraseña
    OR v_Usuario.telefono <> v_telefono
    OR v_Usuario.nombre <> v_nombre
    OR v_Usuario.primerApellido <> v_primerApellido
    OR v_Usuario.segundoApellido <> v_segundoApellido
    OR v_Usuario.dni <> v_dni
    OR v_Usuario.direccion <> v_direccion)
    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;

  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumUsuarios NUMBER := 0;
    v_NumClientes NUMBER := 0;
  BEGIN
      SELECT COUNT(*) INTO v_NumUsuarios FROM USUARIO WHERE OID_Us =
v_OID_Us;
      SELECT COUNT(*) INTO v_NumClientes FROM CLIENTE WHERE OID_Us =
v_OID_Us;
      IF v_NumUsuarios = 0 OR v_NumClientes = 0
        THEN v_Salida:=false;
      else
        DELETE FROM USUARIO WHERE OID_Us = v_OID_Us;
      SELECT COUNT(*) INTO v_NumUsuarios FROM USUARIO WHERE OID_Us =
v_OID_Us;
      SELECT COUNT(*) INTO v_NumClientes FROM CLIENTE WHERE OID_Us =
v_OID_Us;
      IF v_NumUsuarios != 0 OR v_NumClientes != 0
        THEN
          v_Salida   :=false;
      else     v_Salida:=true;
      END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/

-- PRUEBAS GERENTE-----------------------------------------------
------------------------------------------------------------

CREATE OR REPLACE PACKAGE PCK_Gerente
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_email IN USUARIO.EMAIL%TYPE,
v_contraseña IN USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, v_email IN USUARIO.EMAIL%TYPE, v_contraseña IN
USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, salidaEsperada BOOLEAN);
END;
/

CREATE OR REPLACE PACKAGE BODY PCK_Gerente
AS
  v_Salida BOOLEAN := TRUE;
  CURSOR C IS
  SELECT * FROM USUARIO NATURAL JOIN GERENTE;
  v_Usuario C%ROWTYPE;
  PROCEDURE Inicializar AS
  BEGIN
    DELETE FROM USUARIO WHERE tipoUsuario='Gerente';
  END Inicializar;

  PROCEDURE Insertar (nombrePrueba VARCHAR2, v_email IN
USUARIO.EMAIL%TYPE, v_contraseña IN USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN)
  AS
  BEGIN
    crearGerente(v_email,v_telefono,v_contraseña);
    SELECT * INTO v_Usuario FROM Usuario NATURAL JOIN GERENTE WHERE
v_email=Usuario.EMAIL;
    IF (v_Usuario.email <> v_email
    OR v_Usuario.contraseña <> v_contraseña
    OR v_Usuario.telefono <> v_telefono)
    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT WORK;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
```

```
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

  PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, v_email IN USUARIO.EMAIL%TYPE, v_contraseña IN
USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN) AS
  BEGIN
    UPDATE Usuario set EMAIL = v_email, "CONTRASEÑA"=v_contraseña,
TELEFONO=v_telefono
      WHERE OID_Us = v_OID_Us;
    SELECT * INTO v_Usuario FROM Usuario NATURAL JOIN GERENTE WHERE
v_OID_Us=OID_Us;
    IF (v_Usuario.email <> v_email
    OR v_Usuario.contraseña <> v_contraseña
    OR v_Usuario.telefono <> v_telefono)
    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;

  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumUsuarios NUMBER := 0;
    v_NumGerentes NUMBER := 0;
  BEGIN
      SELECT COUNT(*) INTO v_NumUsuarios FROM USUARIO WHERE OID_Us =
v_OID_Us;
      SELECT COUNT(*) INTO v_NumGerentes FROM GERENTE WHERE OID_Us =
v_OID_Us;
      IF v_NumUsuarios = 0 OR v_NumGerentes = 0
        THEN v_Salida:=false;
      else
        DELETE FROM USUARIO WHERE OID_Us = v_OID_Us;
      SELECT COUNT(*) INTO v_NumUsuarios FROM USUARIO WHERE OID_Us =
v_OID_Us;
      SELECT COUNT(*) INTO v_NumGerentes FROM GERENTE WHERE OID_Us =
v_OID_Us;
      IF v_NumUsuarios != 0 OR v_NumGerentes != 0
        THEN
```

```
          v_Salida   :=false;
      else     v_Salida:=true;
      END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/

-- PRUEBAS DEPENDIENTE-----------------------------------------------
------------------------------------------------------------------

    CREATE OR REPLACE PACKAGE PCK_Dependiente
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_email IN USUARIO.EMAIL%TYPE,
v_contraseña IN USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, v_email IN USUARIO.EMAIL%TYPE, v_contraseña IN
USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, salidaEsperada BOOLEAN);
END;
/

CREATE OR REPLACE PACKAGE BODY PCK_Dependiente
AS
  v_Salida BOOLEAN := TRUE;
  CURSOR C IS
  SELECT * FROM USUARIO NATURAL JOIN DEPENDIENTE;
  v_Usuario C%ROWTYPE;
  PROCEDURE Inicializar AS
  BEGIN
    DELETE FROM USUARIO WHERE tipoUsuario ='Dependiente';
  END Inicializar;

  PROCEDURE Insertar (nombrePrueba VARCHAR2, v_email IN
USUARIO.EMAIL%TYPE, v_contraseña IN USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN)
  AS
  BEGIN
    crearDependiente(v_email,v_telefono,v_contraseña);
    SELECT * INTO v_Usuario FROM Usuario NATURAL JOIN DEPENDIENTE WHERE
v_email=Usuario.EMAIL;
```

```
    IF (v_Usuario.email <> v_email
    OR v_Usuario.contraseña <> v_contraseña
    OR v_Usuario.telefono <> v_telefono)
    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT WORK;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

  PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, v_email IN USUARIO.EMAIL%TYPE, v_contraseña IN
USUARIO."CONTRASEÑA"%TYPE,
  v_telefono IN USUARIO.TELEFONO%TYPE, salidaEsperada BOOLEAN) AS
  BEGIN
    UPDATE Usuario set EMAIL = v_email, "CONTRASEÑA"=v_contraseña,
TELEFONO=v_telefono
      WHERE OID_Us = v_OID_Us;
    SELECT * INTO v_Usuario FROM Usuario NATURAL JOIN DEPENDIENTE WHERE
v_OID_Us=OID_Us;
    IF (v_Usuario.email <> v_email
    OR v_Usuario.contraseña <> v_contraseña
    OR v_Usuario.telefono <> v_telefono)
    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;

  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Us IN
Usuario.OID_Us%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumUsuarios NUMBER := 0;
    v_NumDependientes NUMBER := 0;
  BEGIN
      SELECT COUNT(*) INTO v_NumUsuarios FROM USUARIO WHERE OID_Us =
v_OID_Us;
      SELECT COUNT(*) INTO v_NumDependientes FROM DEPENDIENTE WHERE OID_Us
= v_OID_Us;
      IF v_NumUsuarios = 0 OR v_NumDependientes = 0
```

```
        THEN v_Salida:=false;
      else
        DELETE FROM USUARIO WHERE OID_Us = v_OID_Us;
      SELECT COUNT(*) INTO v_NumUsuarios FROM USUARIO WHERE OID_Us =
v_OID_Us;
      SELECT COUNT(*) INTO v_NumDependientes FROM DEPENDIENTE WHERE OID_Us
= v_OID_Us;
      IF v_NumUsuarios != 0 OR v_NumDependientes != 0
        THEN
          v_Salida   :=false;
      else     v_Salida:=true;
      END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/

-- PRUEBAS VALORACION-----------------------------------------------
----------------------------------------------------------------

CREATE OR REPLACE PACKAGE PCK_Valoracion
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_asunto IN
VALORACION.asunto%TYPE, v_descripcion IN VALORACION.descripcion%TYPE,
  v_OID_Us IN VALORACION.OID_Us%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2,v_OID_V IN
VALORACION.OID_V%TYPE, v_asunto IN VALORACION.asunto%TYPE, v_descripcion
IN VALORACION.descripcion%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_V IN
VALORACION.OID_V%TYPE, salidaEsperada BOOLEAN);
END;
/

CREATE OR REPLACE PACKAGE BODY PCK_Valoracion
AS

v_Salida BOOLEAN := TRUE;
v_Valoracion VALORACION%ROWTYPE;

PROCEDURE Inicializar AS
BEGIN
  DELETE FROM VALORACION;
  END Inicializar;
```

```
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_asunto IN
VALORACION.ASUNTO%TYPE, v_descripcion IN VALORACION.DESCRIPCION%TYPE,
v_OID_Us IN VALORACION.OID_US%TYPE, salidaEsperada BOOLEAN) AS
BEGIN
        crearValoracion(v_asunto, v_descripcion, v_OID_Us);
    SELECT * INTO v_Valoracion FROM VALORACION WHERE v_OID_Us = OID_Us;

    IF (v_Valoracion.asunto <> v_asunto
    OR v_Valoracion.descripcion <> v_descripcion
    OR v_Valoracion.OID_Us <> v_OID_Us)

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_V IN
VALORACION.OID_V%TYPE,
  v_asunto IN VALORACION.ASUNTO%TYPE, v_descripcion IN
VALORACION.DESCRIPCION%TYPE,
  salidaEsperada BOOLEAN) AS
BEGIN
    UPDATE VALORACION set asunto=v_asunto, descripcion=v_descripcion,
fechaEnvio=SYSDATE + 1 WHERE OID_V = v_OID_V;

    SELECT * INTO v_Valoracion FROM VALORACION WHERE OID_V = v_OID_V;

    IF (v_Valoracion.asunto <> v_asunto
    OR v_Valoracion.descripcion <> v_descripcion)

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;
```

```
  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_V IN
VALORACION.OID_V%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumValoraciones NUMBER := 0;
  BEGIN
    SELECT COUNT(*) INTO v_NumValoraciones FROM VALORACION WHERE OID_V =
v_OID_V;
      IF v_NumValoraciones = 0 THEN v_Salida:=false;
      else
    DELETE FROM VALORACION WHERE OID_V = v_OID_V;
    SELECT COUNT(*) INTO v_NumValoraciones FROM VALORACION WHERE OID_V =
v_OID_V;
    IF v_NumValoraciones != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/

-- PRUEBAS CITA-----------------------------------------------
------------------------------------------------------------


CREATE OR REPLACE PACKAGE PCK_Cita
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_fechaAcordada IN
CITA.fechaAcordada%TYPE, v_asunto IN CITA.asunto%TYPE, v_descripcion IN
CITA.descripcion%TYPE, v_tipoCita IN CITA.tipoCita%TYPE, v_OID_Us IN
CITA.OID_Us%TYPE,salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2,v_OID_Ci IN CITA.OID_Ci%TYPE,
v_fechaAcordada IN CITA.fechaAcordada%TYPE, v_asunto IN CITA.asunto%TYPE,
v_descripcion IN CITA.descripcion%TYPE, v_tipoCita IN CITA.tipoCita%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Ci IN CITA.OID_Ci%TYPE,
salidaEsperada BOOLEAN);
END;
/


CREATE OR REPLACE PACKAGE BODY PCK_Cita
AS

v_Salida BOOLEAN := TRUE;
```

```
v_Cita CITA%ROWTYPE;

PROCEDURE Inicializar AS
BEGIN
  DELETE FROM CITA;
  END Inicializar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_fechaAcordada IN
CITA.fechaAcordada%TYPE, v_asunto IN CITA.asunto%TYPE, v_descripcion IN
CITA.descripcion%TYPE, v_tipoCita IN CITA.tipoCita%TYPE, v_OID_Us IN
CITA.OID_Us%TYPE,salidaEsperada BOOLEAN) AS
BEGIN
        crearCita(v_fechaAcordada, v_asunto, v_descripcion,
v_tipoCita,v_OID_Us);
    SELECT * INTO v_Cita FROM CITA WHERE v_fechaAcordada = fechaAcordada
AND v_asunto = asunto AND v_descripcion = descripcion AND v_tipoCita =
tipoCita AND v_OID_Us = OID_Us;

    IF (v_Cita.fechaAcordada <> v_fechaAcordada
    OR v_Cita.asunto <> v_asunto
    OR v_Cita.descripcion <> v_descripcion
    OR v_Cita.tipoCita <> v_tipoCita
    OR v_Cita.OID_Us <> v_OID_Us)

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2,v_OID_Ci IN CITA.OID_Ci%TYPE,
v_fechaAcordada IN CITA.fechaAcordada%TYPE, v_asunto IN CITA.asunto%TYPE,
v_descripcion IN CITA.descripcion%TYPE, v_tipoCita IN CITA.tipoCita%TYPE,
salidaEsperada BOOLEAN) AS
BEGIN
    UPDATE CITA set fechaAcordada=v_fechaAcordada, asunto=v_asunto,
descripcion=v_descripcion, tipoCita=v_tipoCita WHERE OID_Ci = v_OID_Ci;

    SELECT * INTO v_Cita FROM CITA WHERE v_OID_Ci = OID_Ci;

    IF (v_Cita.fechaAcordada <> v_fechaAcordada
    OR v_Cita.asunto <> v_asunto
    OR v_Cita.descripcion <> v_descripcion
    OR v_Cita.tipoCita <> v_tipoCita)

    THEN
```

```
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;


  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Ci IN CITA.OID_Ci%TYPE,
salidaEsperada BOOLEAN)
  AS
    v_NumCitas NUMBER := 0;
  BEGIN
    SELECT COUNT(*) INTO v_NumCitas FROM CITA WHERE OID_Ci = v_OID_Ci;
      IF v_NumCitas = 0 THEN v_Salida:=false;
      else
    DELETE FROM CITA WHERE OID_Ci = v_OID_Ci;
    SELECT COUNT(*) INTO v_NumCitas FROM CITA WHERE OID_Ci = v_OID_Ci;
    IF v_NumCitas != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/
-- PRUEBAS PROVEEDOR-----------------------------------------------
----------------------------------------------------------------


CREATE OR REPLACE PACKAGE PCK_Proveedor
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_codigo IN
PROVEEDOR.codigo%TYPE, v_nombre IN PROVEEDOR.nombre%TYPE, v_email IN
PROVEEDOR.email%TYPE, v_telefono IN PROVEEDOR.telefono%TYPE, v_direccion
IN PROVEEDOR.direccion%TYPE, v_web IN PROVEEDOR.web%TYPE, salidaEsperada
BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Prov IN
PROVEEDOR.OID_Prov%TYPE, v_codigo IN PROVEEDOR.codigo%TYPE, v_nombre IN
PROVEEDOR.nombre%TYPE, v_email IN PROVEEDOR.email%TYPE, v_telefono IN
```

```
PROVEEDOR.telefono%TYPE, v_direccion IN PROVEEDOR.direccion%TYPE, v_web IN
PROVEEDOR.web%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Prov IN
PROVEEDOR.OID_Prov%TYPE, salidaEsperada BOOLEAN);
END;
/


CREATE OR REPLACE PACKAGE BODY PCK_Proveedor
AS

v_Salida BOOLEAN := TRUE;
v_Proveedor PROVEEDOR%ROWTYPE;

PROCEDURE Inicializar AS
BEGIN
  DELETE FROM PROVEEDOR;
  END Inicializar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_codigo IN
PROVEEDOR.codigo%TYPE, v_nombre IN PROVEEDOR.nombre%TYPE, v_email IN
PROVEEDOR.email%TYPE, v_telefono IN PROVEEDOR.telefono%TYPE, v_direccion
IN PROVEEDOR.direccion%TYPE, v_web IN PROVEEDOR.web%TYPE, salidaEsperada
BOOLEAN) AS
BEGIN
        crearProveedor(v_codigo, v_nombre, v_email, v_telefono,
v_direccion, v_web);
    SELECT * INTO v_Proveedor FROM PROVEEDOR WHERE v_codigo = codigo;

    IF (v_Proveedor.codigo <> v_codigo
    OR v_Proveedor.nombre <> v_nombre
    OR v_Proveedor.email <> v_email
    OR v_Proveedor.telefono <> v_telefono
    OR v_Proveedor.direccion <> v_direccion
    OR v_Proveedor.web <> v_web)

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Prov IN
PROVEEDOR.OID_Prov%TYPE, v_codigo IN PROVEEDOR.codigo%TYPE, v_nombre IN
PROVEEDOR.nombre%TYPE, v_email IN PROVEEDOR.email%TYPE, v_telefono IN
```

```
PROVEEDOR.telefono%TYPE, v_direccion IN PROVEEDOR.direccion%TYPE, v_web IN
PROVEEDOR.web%TYPE, salidaEsperada BOOLEAN) AS
BEGIN
    UPDATE PROVEEDOR set codigo=v_codigo, nombre=v_nombre, email=v_email,
telefono=v_telefono, direccion=v_direccion, web=v_web WHERE OID_Prov =
v_OID_Prov;

    SELECT * INTO v_Proveedor FROM PROVEEDOR WHERE v_OID_Prov = OID_Prov;

    IF (v_Proveedor.codigo <> v_codigo
    OR v_Proveedor.nombre <> v_nombre
    OR v_Proveedor.email <> v_email
    OR v_Proveedor.telefono <> v_telefono
    OR v_Proveedor.direccion <> v_direccion
    OR v_Proveedor.web <> v_web)

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;


  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Prov IN
PROVEEDOR.OID_Prov%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumProveedores NUMBER := 0;
  BEGIN
    SELECT COUNT(*) INTO v_NumProveedores FROM PROVEEDOR WHERE OID_Prov =
v_OID_Prov;
      IF v_NumProveedores = 0 THEN v_Salida:=false;
      else
    DELETE FROM PROVEEDOR WHERE OID_Prov = v_OID_Prov;
    SELECT COUNT(*) INTO v_NumProveedores FROM PROVEEDOR WHERE OID_Prov =
v_OID_Prov;
    IF v_NumProveedores != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
```

```
        ROLLBACK;
  END Eliminar;
END;
/

-- PRUEBAS PRODUCTOMOTOR----------------------------------------------
------------------------------------------------------------------

CREATE OR REPLACE PACKAGE PCK_ProductoMotor
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_codigo IN
PRODUCTO.codigo%TYPE, v_nombre IN PRODUCTO.nombre%TYPE, v_descripcion IN
PRODUCTO.descripcion%TYPE, v_marca IN PRODUCTO.marca%TYPE,
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_prov%TYPE,
v_cilindrada IN MOTOR.cilindrada%TYPE ,v_estado IN MOTOR.estado%TYPE,
v_anyoFabricacion IN MOTOR.anyoFabricacion%TYPE, v_tipoMotor IN
MOTOR.tipoMotor%TYPE, v_garantia IN MOTOR.garantia%TYPE,salidaEsperada
BOOLEAN);

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE, v_codigo IN PRODUCTO.codigo%TYPE,  v_nombre IN
PRODUCTO.nombre%TYPE, v_descripcion IN PRODUCTO.descripcion%TYPE, v_marca
IN PRODUCTO.marca%TYPE,
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_prov%TYPE,
v_cilindrada IN MOTOR.cilindrada%TYPE ,v_estado IN MOTOR.estado%TYPE,
v_anyoFabricacion IN MOTOR.anyoFabricacion%TYPE, v_tipoMotor IN
MOTOR.tipoMotor%TYPE, v_garantia IN MOTOR.garantia%TYPE,salidaEsperada
BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_P IN PRODUCTO.OID_P%TYPE,
salidaEsperada BOOLEAN);
END;
/


CREATE OR REPLACE PACKAGE BODY PCK_ProductoMotor
AS

v_Salida BOOLEAN := TRUE;

CURSOR C IS
  SELECT * FROM PRODUCTO NATURAL JOIN MOTOR;
  v_Producto C%ROWTYPE;
  PROCEDURE Inicializar AS
  BEGIN
    DELETE FROM PRODUCTO WHERE TipoProducto='Motor';
  END Inicializar;
```

```
PROCEDURE Insertar (nombrePrueba VARCHAR2, v_codigo IN
PRODUCTO.codigo%TYPE, v_nombre IN PRODUCTO.nombre%TYPE, v_descripcion IN
PRODUCTO.descripcion%TYPE, v_marca IN PRODUCTO.marca%TYPE,
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_Prov%TYPE,
v_cilindrada IN MOTOR.cilindrada%TYPE ,v_estado IN MOTOR.estado%TYPE,
v_anyoFabricacion IN MOTOR.anyoFabricacion%TYPE, v_tipoMotor IN
MOTOR.tipoMotor%TYPE, v_garantia IN MOTOR.garantia%TYPE,salidaEsperada
BOOLEAN) AS
BEGIN
        crearMotor(v_codigo ,v_nombre, v_descripcion, v_marca, v_precio,
v_oferta, v_iva, v_stock, v_stockMinimo,V_OID_Prov, v_cilindrada,
v_estado, v_anyoFabricacion, v_tipoMotor, v_garantia);
    SELECT * INTO v_Producto FROM PRODUCTO NATURAL JOIN MOTOR WHERE
v_codigo = codigo;

    IF (v_Producto.codigo <> v_codigo
    OR v_Producto.nombre <> v_nombre
    OR v_Producto.descripcion <> v_descripcion
    OR v_Producto.marca <> v_marca
    OR v_Producto.precio <> v_precio
    OR v_Producto.oferta <> v_oferta
    OR v_Producto.iva <> v_iva
    OR v_Producto.stock <> v_stock
    OR v_Producto.stockMinimo <> v_stockMinimo
    OR v_Producto.OID_Prov <> v_OID_Prov
    OR v_Producto.cilindrada <> v_cilindrada
    OR v_Producto.estado <> v_estado
    OR v_Producto.anyoFabricacion <> v_anyoFabricacion
    OR v_Producto.tipoMotor <> v_tipoMotor
    OR v_Producto.garantia <> v_garantia
    )

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE,v_codigo IN PRODUCTO.codigo%TYPE, v_nombre IN
PRODUCTO.nombre%TYPE, v_descripcion IN PRODUCTO.descripcion%TYPE, v_marca
IN PRODUCTO.marca%TYPE,
```

```
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_prov%TYPE,
v_cilindrada IN MOTOR.cilindrada%TYPE ,v_estado IN MOTOR.estado%TYPE,
v_anyoFabricacion IN MOTOR.anyoFabricacion%TYPE, v_tipoMotor IN
MOTOR.tipoMotor%TYPE, v_garantia IN MOTOR.garantia%TYPE,salidaEsperada
BOOLEAN) AS
BEGIN
    UPDATE PRODUCTO set codigo=v_codigo, nombre=v_nombre,
descripcion=v_descripcion, marca=v_marca, tipoProducto='Motor',
precio=v_precio

,oferta=v_oferta,iva=v_iva,stock=v_stock,stockMinimo=v_stockMinimo,OID_Pro
v=v_OID_Prov WHERE OID_P = v_OID_P;
        UPDATE MOTOR set cilindrada=v_cilindrada, estado=v_estado,
anyoFabricacion=v_anyoFabricacion, tipoMotor=v_tipoMotor,
garantia=v_garantia WHERE OID_P = v_OID_P;

    SELECT * INTO v_Producto FROM PRODUCTO NATURAL JOIN MOTOR WHERE  OID_P
= v_OID_P;

    IF (v_Producto.codigo <> v_codigo
    OR v_Producto.nombre <> v_nombre
    OR v_Producto.descripcion <> v_descripcion
    OR v_Producto.marca <> v_marca
    OR v_Producto.precio <> v_precio
    OR v_Producto.oferta <> v_oferta
    OR v_Producto.iva <> v_iva
    OR v_Producto.stock <> v_stock
    OR v_Producto.stockMinimo <> v_stockMinimo
    OR v_Producto.OID_Prov <> v_OID_Prov
    OR v_Producto.cilindrada <> v_cilindrada
    OR v_Producto.estado <> v_estado
    OR v_Producto.anyoFabricacion <> v_anyoFabricacion
    OR v_Producto.tipoMotor <> v_tipoMotor
    OR v_Producto.garantia <> v_garantia
    )

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;
```

```
  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumProductos NUMBER := 0;
        v_NumMotor NUMBER := 0;
  BEGIN
    SELECT COUNT(*) INTO v_NumProductos FROM PRODUCTO WHERE OID_P =
v_OID_P;
      SELECT COUNT(*) INTO v_NumMotor FROM MOTOR WHERE OID_P = v_OID_P;

      IF v_NumProductos = 0 OR V_NumMotor = 0 THEN v_Salida:=false;
      else
    DELETE FROM PRODUCTO WHERE OID_P = v_OID_P;
    SELECT COUNT(*) INTO v_NumProductos FROM PRODUCTO WHERE OID_P =
v_OID_P;
          SELECT COUNT(*) INTO v_NumMotor FROM MOTOR WHERE OID_P =
v_OID_P;
    IF v_NumProductos != 0 OR v_NumMotor != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/

-- PRUEBAS PRODUCTORECAMBIO-----------------------------------------
-----------------------------------------------------------------

CREATE OR REPLACE PACKAGE PCK_ProductoRecambio
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_codigo IN
PRODUCTO.codigo%TYPE, v_nombre IN PRODUCTO.nombre%TYPE, v_descripcion IN
PRODUCTO.descripcion%TYPE, v_marca IN PRODUCTO.marca%TYPE,
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN
PRODUCTO.OID_prov%TYPE,salidaEsperada BOOLEAN);

PROCEDURE Actualizar (nombrePrueba VARCHAR2,  v_OID_P IN
PRODUCTO.OID_P%TYPE, v_codigo IN PRODUCTO.codigo%TYPE, v_nombre IN
PRODUCTO.nombre%TYPE, v_descripcion IN PRODUCTO.descripcion%TYPE, v_marca
IN PRODUCTO.marca%TYPE,
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
```

```
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN
PRODUCTO.OID_prov%TYPE,salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_P IN PRODUCTO.OID_P%TYPE,
salidaEsperada BOOLEAN);
END;
/


CREATE OR REPLACE PACKAGE BODY PCK_ProductoRecambio
AS

v_Salida BOOLEAN := TRUE;

CURSOR C IS
  SELECT * FROM PRODUCTO NATURAL JOIN RECAMBIO;
  v_Producto C%ROWTYPE;
  PROCEDURE Inicializar AS
  BEGIN
    DELETE FROM PRODUCTO WHERE TipoProducto='Recambio';
  END Inicializar;


PROCEDURE Insertar (nombrePrueba VARCHAR2, v_codigo IN
PRODUCTO.codigo%TYPE, v_nombre IN PRODUCTO.nombre%TYPE, v_descripcion IN
PRODUCTO.descripcion%TYPE, v_marca IN PRODUCTO.marca%TYPE,
  v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE,
v_iva IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE,
  v_stockMinimo IN PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN
PRODUCTO.OID_prov%TYPE,salidaEsperada BOOLEAN) AS
BEGIN
        crearRecambio(v_codigo, v_nombre, v_descripcion, v_marca,
v_precio, v_oferta, v_iva, v_stock, v_stockMinimo,V_OID_Prov);
    SELECT * INTO v_Producto FROM PRODUCTO NATURAL JOIN RECAMBIO WHERE
v_codigo = codigo;

    IF (v_Producto.codigo <> v_codigo
    OR v_Producto.nombre <> v_nombre
    OR v_Producto.descripcion <> v_descripcion
    OR v_Producto.marca <> v_marca
    OR v_Producto.precio <> v_precio
    OR v_Producto.oferta <> v_oferta
    OR v_Producto.iva <> v_iva
    OR v_Producto.stock <> v_stock
    OR v_Producto.stockMinimo <> v_stockMinimo
    OR v_Producto.OID_Prov <> v_OID_Prov
    )

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
```

```
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE, v_codigo IN PRODUCTO.codigo%TYPE, v_nombre IN
PRODUCTO.nombre%TYPE, v_descripcion IN PRODUCTO.descripcion%TYPE, v_marca
IN PRODUCTO.marca%TYPE,
  v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE,
v_iva IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo
IN PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN
PRODUCTO.OID_prov%TYPE,salidaEsperada BOOLEAN) AS
BEGIN
    UPDATE PRODUCTO set codigo=v_codigo, nombre=v_nombre,
descripcion=v_descripcion, marca=v_marca,
tipoProducto='Recambio',precio=v_precio,
    oferta=v_oferta, iva=v_iva,stock=v_stock, stockMinimo=v_stockMinimo,
OID_Prov=v_OID_Prov WHERE OID_P = v_OID_P;

    SELECT * INTO v_Producto FROM PRODUCTO NATURAL JOIN RECAMBIO WHERE
OID_P = v_OID_P;

    IF (v_Producto.codigo <> v_codigo
    OR v_Producto.nombre <> v_nombre
    OR v_Producto.descripcion <> v_descripcion
    OR v_Producto.marca <> v_marca
    OR v_Producto.precio <> v_precio
    OR v_Producto.oferta <> v_oferta
    OR v_Producto.iva <> v_iva
    OR v_Producto.stock <> v_stock
    OR v_Producto.stockMinimo <> v_stockMinimo
    OR v_Producto.OID_Prov <> v_OID_Prov
    )

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;
```

```
 PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumProductos NUMBER := 0;
        v_NumRecambios NUMBER := 0;
  BEGIN
    SELECT COUNT(*) INTO v_NumProductos FROM PRODUCTO WHERE OID_P =
v_OID_P;
      SELECT COUNT(*) INTO V_NumRecambios FROM RECAMBIO WHERE OID_P =
v_OID_P;

      IF v_NumProductos = 0 OR V_NumRecambios = 0 THEN v_Salida:=false;
      else
    DELETE FROM PRODUCTO WHERE OID_P = v_OID_P;
    SELECT COUNT(*) INTO v_NumProductos FROM PRODUCTO WHERE OID_P =
v_OID_P;
          SELECT COUNT(*) INTO v_NumRecambios FROM RECAMBIO WHERE OID_P =
v_OID_P;
    IF v_NumProductos != 0 OR v_NumRecambios != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/


-- PRUEBAS PRODUCTOEQUIPAMIENTO----------------------------------------
--
-------------------------------------------------------------------


CREATE OR REPLACE PACKAGE PCK_ProductoEquipamiento
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_codigo IN
PRODUCTO.codigo%TYPE, v_nombre IN PRODUCTO.nombre%TYPE, v_descripcion IN
PRODUCTO.descripcion%TYPE, v_marca IN PRODUCTO.marca%TYPE,
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_prov%TYPE,
v_color IN EQUIPAMIENTO.color%TYPE ,v_material IN
EQUIPAMIENTO.material%TYPE, v_talla IN
EQUIPAMIENTO.talla%TYPE,salidaEsperada BOOLEAN);
```

```
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE, v_codigo IN PRODUCTO.codigo%TYPE, v_nombre IN
PRODUCTO.nombre%TYPE, v_descripcion IN PRODUCTO.descripcion%TYPE, v_marca
IN PRODUCTO.marca%TYPE,
v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE, v_iva
IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo IN
PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_prov%TYPE,
v_color IN EQUIPAMIENTO.color%TYPE ,v_material IN
EQUIPAMIENTO.material%TYPE, v_talla IN
EQUIPAMIENTO.talla%TYPE,salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_P IN PRODUCTO.OID_P%TYPE,
salidaEsperada BOOLEAN);
END;
/


CREATE OR REPLACE PACKAGE BODY PCK_ProductoEquipamiento
AS

v_Salida BOOLEAN := TRUE;

CURSOR C IS
  SELECT * FROM PRODUCTO NATURAL JOIN EQUIPAMIENTO;
  v_Producto C%ROWTYPE;
  PROCEDURE Inicializar AS
  BEGIN
    DELETE FROM PRODUCTO WHERE TipoProducto = 'Equipamiento';
  END Inicializar;


PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_codigo IN
PRODUCTO.codigo%TYPE, v_nombre IN PRODUCTO.nombre%TYPE, v_descripcion IN
PRODUCTO.descripcion%TYPE, v_marca IN PRODUCTO.marca%TYPE,
  v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE,
v_iva IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo
IN PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_Prov%TYPE,
  v_color IN EQUIPAMIENTO.color%TYPE ,v_material IN
EQUIPAMIENTO.material%TYPE, v_talla IN
EQUIPAMIENTO.talla%TYPE,salidaEsperada BOOLEAN) AS
BEGIN
        crearEquipamiento(v_codigo,v_nombre, v_descripcion, v_marca,
v_precio, v_oferta, v_iva, v_stock,
v_stockMinimo,V_OID_Prov,v_color,v_material,v_talla);
    SELECT * INTO v_Producto FROM PRODUCTO NATURAL JOIN EQUIPAMIENTO WHERE
v_codigo = codigo;

    IF (v_Producto.codigo <> v_codigo
    OR v_Producto.nombre <> v_nombre
    OR v_Producto.descripcion <> v_descripcion
    OR v_Producto.marca <> v_marca
    OR v_Producto.precio <> v_precio
    OR v_Producto.oferta <> v_oferta
```

```
    OR v_Producto.iva <> v_iva
    OR v_Producto.stock <> v_stock
    OR v_Producto.stockMinimo <> v_stockMinimo
    OR v_Producto.OID_Prov <> v_OID_Prov
    OR v_Producto.color <> v_color
    OR v_Producto.material <> v_material
    OR v_Producto.talla <> v_talla
    )

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE, v_codigo IN PRODUCTO.codigo%TYPE, v_nombre IN
PRODUCTO.nombre%TYPE, v_descripcion IN PRODUCTO.descripcion%TYPE, v_marca
IN PRODUCTO.marca%TYPE,
  v_precio IN PRODUCTO.precio%TYPE, v_oferta IN PRODUCTO.oferta%TYPE,
v_iva IN PRODUCTO.iva%TYPE, v_stock IN PRODUCTO.stock%TYPE, v_stockMinimo
IN PRODUCTO.stockMinimo%TYPE,v_OID_Prov IN PRODUCTO.OID_prov%TYPE,
  v_color IN EQUIPAMIENTO.color%TYPE ,v_material IN
EQUIPAMIENTO.material%TYPE, v_talla IN
EQUIPAMIENTO.talla%TYPE,salidaEsperada BOOLEAN) AS
BEGIN
    UPDATE PRODUCTO set codigo = v_codigo, nombre=v_nombre,
descripcion=v_descripcion, marca=v_marca,tipoProducto='Equipamiento',
precio=v_precio

,oferta=v_oferta,iva=v_iva,stock=v_stock,stockMinimo=v_stockMinimo,OID_Pro
v=v_OID_Prov WHERE OID_P = v_OID_P;
    UPDATE EQUIPAMIENTO set
color=v_color,talla=v_talla,material=v_material WHERE OID_P=v_OID_P;

    SELECT * INTO v_Producto FROM PRODUCTO NATURAL JOIN EQUIPAMIENTO WHERE
v_OID_P = OID_P;

    IF (v_Producto.codigo <> v_codigo
    OR v_Producto.nombre <> v_nombre
    OR v_Producto.descripcion <> v_descripcion
    OR v_Producto.marca <> v_marca
    OR v_Producto.precio <> v_precio
    OR v_Producto.oferta <> v_oferta
    OR v_Producto.iva <> v_iva
    OR v_Producto.stock <> v_stock
```

```
    OR v_Producto.stockMinimo <> v_stockMinimo
    OR v_Producto.OID_Prov <> v_OID_Prov
     OR v_Producto.color <> v_color
    OR v_Producto.material <> v_material
    OR v_Producto.talla <> v_talla
    )

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;


  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_P IN
PRODUCTO.OID_P%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumProductos NUMBER := 0;
        V_NumEquipamientos NUMBER := 0;
  BEGIN
    SELECT COUNT(*) INTO v_NumProductos FROM PRODUCTO WHERE OID_P =
v_OID_P;
      SELECT COUNT(*) INTO V_NumEquipamientos FROM EQUIPAMIENTO WHERE
OID_P = v_OID_P;

      IF v_NumProductos = 0 OR V_NumEquipamientos = 0 THEN
v_Salida:=false;
      else
    DELETE FROM PRODUCTO WHERE OID_P = v_OID_P;
    SELECT COUNT(*) INTO v_NumProductos FROM PRODUCTO WHERE OID_P =
v_OID_P;
          SELECT COUNT(*) INTO v_NumEquipamientos FROM EQUIPAMIENTO WHERE
OID_P = v_OID_P;
    IF v_NumProductos != 0 OR V_NumEquipamientos != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
```

```
END;
/


-- PRUEBAS PEDIDO-----------------------------------------
------------------------------------------------------------



CREATE OR REPLACE PACKAGE PCK_Pedido
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_numeroPedido IN
PEDIDO.numeroPedido%TYPE, v_fechaEntrega IN PEDIDO.fechaEntrega%TYPE,
v_envio IN PEDIDO.envio%TYPE,
v_tipoPago IN PEDIDO.tipoPago%TYPE,v_OID_Us IN PEDIDO.OID_Us%TYPE
,salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Pe IN
PEDIDO.OID_Pe%TYPE, v_numeroPedido IN PEDIDO.numeroPedido%TYPE,
v_fechaEntrega IN PEDIDO.fechaEntrega%TYPE, v_envio IN PEDIDO.envio%TYPE,
v_tipoPago IN PEDIDO.tipoPago%TYPE, v_estado IN PEDIDO.estado%TYPE
,salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Pe IN PEDIDO.OID_Pe%TYPE,
salidaEsperada BOOLEAN);
END;
/


CREATE OR REPLACE PACKAGE BODY PCK_Pedido
AS

v_Salida BOOLEAN := TRUE;
v_Pedido PEDIDO%ROWTYPE;

PROCEDURE Inicializar AS
BEGIN
  DELETE FROM PEDIDO;
  END Inicializar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_numeroPedido IN
PEDIDO.numeroPedido%TYPE, v_fechaEntrega IN PEDIDO.fechaEntrega%TYPE,
v_envio IN PEDIDO.envio%TYPE,
v_tipoPago IN PEDIDO.tipoPago%TYPE,v_OID_Us IN PEDIDO.OID_Us%TYPE
,salidaEsperada BOOLEAN) AS
BEGIN
        crearPedido(v_numeroPedido, v_fechaEntrega, v_envio,
v_tipoPago,v_OID_Us);
    SELECT * INTO v_Pedido FROM PEDIDO WHERE v_numeroPedido =
numeroPedido;

    IF (v_Pedido.numeroPedido <> v_numeroPedido
    OR v_Pedido.fechaEntrega <> v_fechaEntrega
    OR v_Pedido.envio <> v_envio
```

```
    OR v_Pedido.tipoPago <> v_tipoPago
    OR v_Pedido.OID_Us <> v_OID_Us)

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v_OID_Pe IN
PEDIDO.OID_Pe%TYPE, v_numeroPedido IN PEDIDO.numeroPedido%TYPE,
v_fechaEntrega IN PEDIDO.fechaEntrega%TYPE, v_envio IN PEDIDO.envio%TYPE,
v_tipoPago IN PEDIDO.tipoPago%TYPE, v_estado IN PEDIDO.estado%TYPE
,salidaEsperada BOOLEAN) AS
BEGIN
    UPDATE PEDIDO set numeroPedido=v_numeroPedido,
fechaEntrega=v_fechaEntrega, envio=v_envio,tipoPago=v_tipoPago,
estado=v_estado WHERE OID_Pe = v_OID_Pe;

    SELECT * INTO v_Pedido FROM PEDIDO WHERE v_OID_Pe = OID_Pe;

    IF (v_Pedido.numeroPedido <> v_numeroPedido
    OR v_Pedido.fechaEntrega <> v_fechaEntrega
    OR v_Pedido.envio <> v_envio
    OR v_Pedido.tipoPago <> v_tipoPago
    OR v_Pedido.estado <> v_estado)


    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;


  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_Pe IN
PEDIDO.OID_Pe%TYPE, salidaEsperada BOOLEAN)
  AS
    v_NumPedidos NUMBER := 0;
```

```
  BEGIN
    SELECT COUNT(*) INTO v_NumPedidos FROM PEDIDO WHERE OID_Pe = v_OID_Pe;
      IF v_NumPedidos = 0 THEN v_Salida:=false;
      else
    DELETE FROM PEDIDO WHERE OID_Pe = v_OID_Pe;
    SELECT COUNT(*) INTO v_NumPedidos FROM PEDIDO WHERE OID_Pe = v_OID_Pe;
    IF v_NumPedidos != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/


-- PRUEBAS LINEAPEDIDO-----------------------------------------
------------------------------------------------------------



CREATE OR REPLACE PACKAGE PCK_LineaPedido
AS
PROCEDURE Inicializar;
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_cantidad IN
LINEAPEDIDO.CANTIDAD%TYPE, v_OID_P IN LINEAPEDIDO.OID_P%TYPE,
  v_OID_Pe IN LINEAPEDIDO.OID_PE%TYPE, salidaEsperada BOOLEAN);
PROCEDURE Actualizar (nombrePrueba VARCHAR2,  v_cantidad IN
LINEAPEDIDO.CANTIDAD%TYPE, v_OID_LP IN LINEAPEDIDO.OID_LP%TYPE,
salidaEsperada BOOLEAN);
PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_LP IN
LINEAPEDIDO.OID_LP%TYPE, salidaEsperada BOOLEAN);
END;
/


CREATE OR REPLACE PACKAGE BODY PCK_LineaPedido
AS

v_Salida BOOLEAN := TRUE;
v_LineaPedido LineaPedido%ROWTYPE;

PROCEDURE Inicializar AS
BEGIN
  DELETE FROM LINEAPEDIDO;
  END Inicializar;
```

```
PROCEDURE Insertar (nombrePrueba VARCHAR2,  v_cantidad IN
LINEAPEDIDO.CANTIDAD%TYPE, v_OID_P IN LINEAPEDIDO.OID_P%TYPE,
  v_OID_Pe IN LINEAPEDIDO.OID_PE%TYPE, salidaEsperada BOOLEAN) AS
BEGIN
        crearLineaPedido(v_cantidad,v_OID_P, v_OID_Pe);
    SELECT * INTO v_LineaPedido FROM LINEAPEDIDO WHERE v_OID_Pe = OID_Pe
AND v_OID_P=OID_P;

    IF (v_LineaPedido.cantidad <> v_cantidad
    OR v_LineaPedido.OID_P <> v_OID_P
    OR v_LineaPedido.OID_Pe <> v_OID_Pe)

    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2,  v_cantidad IN
LINEAPEDIDO.CANTIDAD%TYPE, v_OID_LP IN LINEAPEDIDO.OID_LP%TYPE,
salidaEsperada BOOLEAN) AS
BEGIN
    UPDATE LINEAPEDIDO set cantidad=v_cantidad WHERE OID_LP = v_OID_LP;

    SELECT * INTO v_LineaPedido FROM LINEAPEDIDO WHERE v_OID_LP = OID_LP;

    IF (v_LineaPedido.cantidad <> v_cantidad)


    THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Actualizar;


  PROCEDURE Eliminar (nombrePrueba VARCHAR2, v_OID_LP IN
LINEAPEDIDO.OID_LP%TYPE, salidaEsperada BOOLEAN)
```

```
  AS
    v_NumLineasPedidos NUMBER := 0;
  BEGIN
    SELECT COUNT(*) INTO v_NumLineasPedidos FROM LINEAPEDIDO WHERE OID_LP
= v_OID_LP;
      IF v_NumLineasPedidos = 0 THEN v_Salida:=false;
      else
    DELETE FROM LINEAPEDIDO WHERE OID_LP = v_OID_LP;
    SELECT COUNT(*) INTO v_NumLineasPedidos FROM LINEAPEDIDO WHERE OID_LP
= v_OID_LP;
    IF v_NumLineasPedidos != 0 THEN
      v_Salida   :=false;   else     v_Salida:=true;
    END IF;
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(v_Salida,
salidaEsperada));
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(nombrePrueba || ': ' || ASSERT_EQUALS(FALSE,
salidaEsperada));
        ROLLBACK;
  END Eliminar;
END;
/
```

6. Script de pruebas

```
SET SERVEROUTPUT ON;
DROP SEQUENCE sec_usuario;
DROP SEQUENCE sec_valoracion;
DROP SEQUENCE sec_cita;
DROP SEQUENCE sec_pedido;
DROP SEQUENCE sec_lineaPedido;
DROP SEQUENCE sec_producto;
DROP SEQUENCE sec_proveedor;
DROP SEQUENCE sec_equipamiento;

CREATE SEQUENCE sec_usuario INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_valoracion INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_cita INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_pedido INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_lineaPedido INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_producto INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_proveedor INCREMENT BY 1 START WITH 1;
CREATE SEQUENCE sec_equipamiento INCREMENT BY 1 START WITH 1;
```

```
--PRUEBAS PROVEEDOR----------------------------------------
-------------------------------------------------------------
BEGIN
PCK_Proveedor.Inicializar;
PCK_Proveedor.Insertar('Insertar un proveedor correctamente','5000000',
'Susuki', 'PedroPerez@Susuki.es','648456571','Gran via
34','www.suzuki.es',TRUE);
PCK_Proveedor.Insertar('Insertar un proveedor correctamente','5000001',
'Honda', 'empresaHonda@gmail.es','648456572','Luca Ferrer
50','www.honda.es',TRUE);
PCK_Proveedor.Insertar('Insertar un proveedor con mismo código','5000001',
'Yamaha', 'PedroPerez@Susuki.es','648456571','Gran via
34','www.suzuki.es',FALSE);
PCK_Proveedor.Insertar('Insertar un proveedor sin código',NULL, 'Yamaha',
'PedroPerez@Yamaha.es','648456573','Gran via 34','www.suzuki.es',FALSE);
PCK_Proveedor.Insertar('Insertar un proveedor sin nombre','5578526', NULL,
'PedroPerez@gmail.es','648456574','Gran via 38','www.yamaha.es',FALSE);
PCK_Proveedor.Insertar('Insertar un proveedor con mismo email','7854788',
'Susuki', 'PedroPerez@Susuki.es','648456571','Gran via
34','www.suzuki.es',FALSE);
PCK_Proveedor.Actualizar('Actualizar proveedor 1',1,'5000000', 'Susuki',
'PedroActuzalizado@Susuki.es','648456571','Gran via
34','www.suzuki.es',TRUE);
PCK_Proveedor.Eliminar('Eliminar un proveedor 2',2,TRUE);
END;
/

--PRUEBAS PRODUCTORECAMBIO----------------------------------
-------------------------------------------------------------
BEGIN
PCK_ProductoRecambio.Inicializar;
PCK_ProductoRecambio.Insertar('Insertar un producto recambio
correctamente','2000000','Bujia', 'Son muy
eficientes','Honda',5.21,4.8,0.21,55,5,1,TRUE);
PCK_ProductoRecambio.Insertar('Insertar un producto recambio
correctamente','2000001','Valvula',
'Inoxidables','Suzuki',48.9,0.0,0.21,31,2,1,TRUE);
PCK_ProductoRecambio.Insertar('Insertar un producto recambio
correctamente','2000003','Escape',
'Sonoro','Yamaha',115.75,null,0.21,31,2,1,TRUE);
PCK_ProductoRecambio.Insertar('Insertar un producto recambio con mismo
código','2000000','Bujia2', 'No son muy
eficientes','Sthil',5.21,4.8,0.21,55,5,1,FALSE);
PCK_ProductoRecambio.Insertar('Insertar un producto recambio sin
código',null,'Recambio1','Son muy
eficientes','Sthil',5.21,1.4,0.21,55,5,1,FALSE);
PCK_ProductoRecambio.Insertar('Insertar un producto recambio sin
nombre','2000001', null,'Son muy
eficientes','Sthil',5.21,1.4,0.21,55,5,1,FALSE);
PCK_ProductoRecambio.Actualizar('Actualizar un producto
recambio',1,'2000000','Bujia', 'Son muy eficientes
actualizada','Honda',5.21,4.8,0.21,55,5,1,TRUE);
```

```
PCK_ProductoRecambio.Eliminar('Eliminar un producto recambio',3,TRUE);
END;
/

--PRUEBAS PRODUCTOMOTOR--------------------------------
------------------------------------------------------------
BEGIN
PCK_ProductoMotor.Inicializar;
PCK_ProductoMotor.Insertar('Insertar un producto motor
correctamente','3000000','Yamaha M1', 'Corre
mucho','Yamaha',19800.21,null,0.21,1,1,1,'900','Nuevo',1990,'Moto',5,TRUE)
;
PCK_ProductoMotor.Insertar('Insertar un producto motor
correctamente','3000001','Motosierra', 'Corta
mucho','Stihl',412,null,0.21,1,1,1,'900','Nuevo',1990,'Moto',2,TRUE);
PCK_ProductoMotor.Insertar('Insertar un producto motor
correctamente','3000002','Kawasaki ninja', 'Corre
poco','Kawasaki',16109.21,null,0.21,1,1,1,'800','Nuevo',1995,'Moto',4,TRUE
);
PCK_ProductoMotor.Insertar('Insertar un producto motor con codigo
repetido','3000001','Bujia', 'Son muy
eficientes','Sthil',5.21,1.4,1.21,55,5,1,'900','Nuevo',1990,'Moto',5,FALSE
);
PCK_ProductoMotor.Insertar('Insertar un producto motor sin
codigo',null,'Motosierra', 'Son muy
eficientes','Sthil',5.21,1.4,1.21,55,5,1,'900','Nuevo',1990,'Moto',5,FALSE
);
PCK_ProductoMotor.Insertar('Insertar un producto motor sin
precio','3000003','Bujia', 'Son muy
eficientes','Sthil',null,1.4,1.21,55,5,1,'900','Nuevo',1990,'Moto',5,FALSE
);
PCK_ProductoMotor.Insertar('Insertar un producto motor sin
iva','3000004','Bujia', 'Son muy
eficientes','Sthil',5.69,1.4,null,55,5,1,'900','Nuevo',1990,'Moto',5,FALSE
);
PCK_ProductoMotor.Insertar('Insertar un producto motor sin
stock','3000005','Bujia', 'Son muy
eficientes','Sthil',5.40,1.4,1.21,null,5,1,'900','Nuevo',1990,'Moto',5,FAL
SE);
PCK_ProductoMotor.Insertar('Insertar un producto motor sin proveedor
asociado','3000006','Bujia', 'Son muy
eficientes','Sthil',5.21,1.4,1.21,55,5,null,'900','Nuevo',1990,'Moto',5,FA
LSE);
PCK_ProductoMotor.Actualizar('Actualizar un producto
motor',8,'3000001','Motosierra', 'Corta
menos','Stihl',412,410,0.21,1,1,1,'900','Nuevo',1990,'Moto',2,TRUE);
PCK_ProductoMotor.Eliminar('Eliminar un producto',9,TRUE);
END;
/

--PRUEBAS PRODUCTOEQUIPAMIENTO--------------------------------
------------------------------------------------------------
BEGIN
```

```
PCK_ProductoEquipamiento.Inicializar;
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento
correcto','4000000','Guantes', 'Son muy
eficientes','AGV',5.21,null,0.21,7,2,1,'Rojo','Algodon','Unica',TRUE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento
correcto','4000001','Casco', 'Es como el de
Rossi','VR46',165.5,null,0.21,2,1,1,'Amarillo','Carton','Unica',TRUE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento
correcto','4000002','Rodilleras', 'Son muy
protectoras','AGV',5.21,1.4,1.21,55,5,1,'Rojo','Fibra','XL',TRUE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento con
el codigo repetido','4000001','Guantes', 'Son muy
eficientes','AGV',5.21,1.4,1.21,55,5,1,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento sin
el codigo',null,'Guantes', 'Son muy
eficientes','AGV',5.21,1.4,1.21,55,5,1,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento sin
el nombre','4000003',null, 'Son muy
eficientes','AGV',5.21,1.4,1.21,55,5,1,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento sin
la descripcion','4000004','Guantes',
null,'AGV',5.21,1.4,1.21,55,5,1,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento sin
precio','4000005','Guantes', 'Son muy
eficientes','AGV',null,1.4,1.21,55,5,1,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento sin
iva','4000006','Guantes', 'Son muy
eficientes','AGV',8.63,1.4,null,55,5,1,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento sin
stock','4000007','Guantes', 'Son muy
eficientes','AGV',5.40,1.4,1.21,null,3,1,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Insertar('Insertar un producto equipamiento sin
proveedor asociado','4000008','Guantes', 'Son muy
eficientes','AGV',5.21,1.4,1.21,55,5,null,'Rojo','Algodon','Unica',FALSE);
PCK_ProductoEquipamiento.Actualizar('Actualizar un producto
equipamiento',17,'4000001','Casco', 'Es como el de
Rossi','VR46',165.5,150,0.21,2,1,1,'Amarillo','Carton','Unica',TRUE);
PCK_ProductoEquipamiento.Eliminar('Eliminar un producto
equipamiento',18,TRUE);
END;
/

--PRUEBAS CLIENTE--------------------------------
------------------------------------------------------------

BEGIN
PCK_Cliente.Inicializar;
PCK_Cliente.Insertar ('Insertar un cliente
valido','sergio@gmail.com','contraseña1','959952885','Sergio','Morales','M
oreno','45877543H','c/Trillo n7', TRUE);
PCK_Cliente.Insertar ('Insertar un cliente
valido','luis@gmail.com','contraseña2','636952885','Luis','Cande','Moreno'
,'33377543H','c/Erfe n7', TRUE);
```

```
PCK_Cliente.Insertar ('Insertar un cliente
valido','LRV@gmail.com','contraseña3','625301566','Luis','Rodríguez','VIdo
sa','45111679J','c/Palos n7', TRUE);
PCK_Cliente.Insertar ('Insertar un cliente con DNI
repetido','respe@gmail.com','contraseña2','959952884','Respe','Rodriguez',
'Valencia','45877543H','c/Trillo n8', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente con contraseña solo de
numeros','pedro@gmail.com','99999999999','959959985','Pedro','Mora','Moro'
,'45920312L','c/Falsa n123', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente con contraseña solo de
letras','pedro@gmail.com','contraseña','959959985','Pedro','Mora','Moro','
45920312L','c/Falsa n123', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente sin
EMAIL',NULL,'contraseña5','959959985','Pedro','Mora','Moro','45920312L','c
/Falsa n123', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente sin
TELEFONO','pedro@gmail.com','contraseña5',NULL,'Pedro','Mora','Moro','4592
0312L','c/Falsa n123', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente sin
NOMBRE','pedro@gmail.com','contraseña5','959959985',NULL,'Mora','Moro','45
920312L','c/Falsa n123', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente sin
APELLIDO','pedro@gmail.com','contraseña5','959959985','Pedro',NULL,NULL,'4
5920312L','c/Falsa n123', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente sin
DNI','pedro@gmail.com','contraseña5','959959985','Pedro','Mora','Moro',NUL
L,'c/Falsa n123', FALSE);
PCK_Cliente.Insertar ('Insertar un cliente sin
DIRECCION','pedro@gmail.com','contraseña5','959959985','Pedro','Mora','Mor
o','45920312L',NULL, FALSE);
PCK_Cliente.Insertar ('Insertar un cliente con menos numeros en el
telefono','sergio@gmail.com','contraseña1','95995288','Juan','Lopez','More
no','45877527H','c/Tarfia n7', FALSE);
PCK_Cliente.Actualizar ('Actualizar un
cliente',1,'sergiomm995@gmail.com','contraseña1','959952885','Sergio','Mor
ales','Moreno','45877543H','c/Trillo n7', TRUE);
PCK_Cliente.Actualizar ('Actualizar un cliente que no
existe',666,'sergiomm995@gmail.com','contraseña1','959952885','Sergio','Mo
rales','Moreno','45877543H','c/Trillo n7', FALSE);
PCK_Cliente.Actualizar ('Actualizar un cliente con contraseña
invalida',1,'NULL','contraseña','959952885','Sergio','Morales','Moreno','4
5877543H','c/Trillo n7', FALSE);
PCK_Cliente.Eliminar ('Eliminar un cliente',2, TRUE);
PCK_Cliente.Eliminar ('Eliminar un cliente que no existe',666, FALSE);
END;
/

--PRUEBAS GERENTE--------------------------------
------------------------------------------------------------
BEGIN
PCK_Gerente.Inicializar;
PCK_Gerente.Insertar ('Insertar un gerente
valido','gerente1@gmail.com','gerente1','610952885', TRUE);
```

```
PCK_Gerente.Insertar ('Insertar un gerente
valido','gerente2@gmail.com','gerente2','301952885', TRUE);
PCK_Gerente.Insertar ('Insertar un gerente
valido','gerenteborrin@gmail.com','borrar2','959252525', TRUE);
PCK_Gerente.Insertar ('Insertar un gerente con TELEFONO
REPETIDO','gerente4@gmail.com','contraseña1','610952885', FALSE);
PCK_Gerente.Insertar ('Insertar un gerente con EMAIL
REPETIDO','gerente1@gmail.com','contraseña1','602132132', FALSE);
PCK_Gerente.Actualizar ('Actualizar un
gerente',15,'gerente2@gmail.com','gerenteActualizado1','301952885', TRUE);
PCK_Gerente.Eliminar ('Eliminar un gerente',16, TRUE);
END;
/


--PRUEBAS DEPENDIENTE-------------------------------
-------------------------------------------------------------
BEGIN
PCK_Dependiente.Inicializar;
PCK_Dependiente.Insertar ('Insertar un dependiente
valido','dependiente1@gmail.com','dependiente1','600952885', TRUE);
PCK_Dependiente.Insertar ('Insertar un dependiente
valido','dependiente2@gmail.com','dependiente2','309952885', TRUE);
PCK_Dependiente.Insertar ('Insertar un dependiente
valido','depenBorrar@gmail.com','borron2','123456789', TRUE);
PCK_Dependiente.Insertar ('Insertar un dependiente
repetido','dependiente1@gmail.com','dependiente1','600952885', FALSE);
PCK_Dependiente.Insertar ('Insertar un dependiente con email
repetido','dependiente1@gmail.com','contraseña40','959952885', FALSE);
PCK_Dependiente.Actualizar ('Actualizar un
dependiente',20,'dependiente2@gmail.com','dependiente2Actualizado','309952
885', TRUE);
PCK_Dependiente.Actualizar ('Actualizar un gerente en la clase
dependiente',15,'gerente2@gmail.com','gerenteActualizado1','301952885',
FALSE);
PCK_Dependiente.Eliminar ('Eliminar un dependiente',21, TRUE);
END;
/

--PRUEBAS PEDIDO-----------------------------------------
-------------------------------------------------------------
BEGIN
PCK_Pedido.Inicializar;
PCK_Pedido.Insertar('Insertar un pedido
correctamente','1234567',to_date('01/04/2019','DD/MM/YYYY'),'True','Tarjet
a',1,TRUE);
PCK_Pedido.Insertar('Insertar un pedido
correctamente','2134568',to_date('10/04/2018','DD/MM/YYYY'),'True','Tarjet
a',1,TRUE);
PCK_Pedido.Insertar('Insertar un pedido
correctamente','1234569',to_date('02/04/2017','DD/MM/YYYY'),'True','Tarjet
a',1,TRUE);
```

```
PCK_Pedido.Insertar('Insertar un pedido sin numero de
pedido',null,to_date('01/04/2019','DD/MM/YYYY'),'True','Tarjeta',1,FALSE);
PCK_Pedido.Insertar('Insertar un pedido sin fecha de
entrega','1234567',null,'True','Tarjeta',1,FALSE);
PCK_Pedido.Insertar('Insertar un pedido con el mismo numero de
pedido','1234567',to_date('01/04/2019','DD/MM/YYYY'),'True','Tarjeta',1,FA
LSE);
PCK_Pedido.Actualizar('Actualizar un
pedido',1,'1234567',to_date('09/06/2018','DD/MM/YYYY'),'False','Tarjeta','
Espera',TRUE);
PCK_Pedido.Eliminar('Eliminar un pedido',2,TRUE);
END;
/



--PRUEBAS CITA-----------------------------------------
----------------------------------------------------
BEGIN
PCK_Cita.Inicializar;
PCK_Cita.Insertar('Insertar una cita
correctamente',to_date('04/09/2013','DD/MM/YYYY'),'Probar moto','Probar
moto Honda','Tienda',1,TRUE);
PCK_Cita.Insertar('Insertar una cita sin una fecha acordada',null,'Probar
moto','Probar moto Honda','Tienda',1,FALSE);
PCK_Cita.Insertar('Insertar una cita sin
asunto',to_date('04/09/2013','DD/MM/YYYY'),null,'Probar moto
Honda','Tienda',1,FALSE);
PCK_Cita.Insertar('Insertar una cita sin
descripcion',to_date('04/09/2013','DD/MM/YYYY'),'Probar
moto',null,'Tienda',1,FALSE);
PCK_Cita.Insertar('Insertar una cita sin cliente
asociado',to_date('04/09/2013','DD/MM/YYYY'),'Probar moto','Probar moto
Honda','Tienda',null,FALSE);
PCK_Cita.Actualizar('Actualizar una
cita',1,to_date('01/04/2019','DD/MM/YYYY'),'Probar moto','Probar moto
Yamaha','Taller',TRUE);
PCK_Cita.Eliminar('Eliminar una cita',1,TRUE);
END;
/

--PRUEBAS VALORACION-----------------------------------------
------------------------------------------------------------
BEGIN
PCK_Valoracion.Inicializar;
PCK_Valoracion.Insertar('Insertar una valoracion
correctamente','Satisfecho', 'Una pasada',1,TRUE);
PCK_Valoracion.Insertar('Insertar una valoracion sin asunto',null, 'Una
pasada',1,FALSE);
PCK_Valoracion.Insertar('Insertar una valoracion sin
descripcion','Satisfecho', null,1,FALSE);
PCK_Valoracion.Insertar('Insertar una valoracion sin cliente
asociado','Satisfecho','Una pasada',null,FALSE);
```

```
PCK_Valoracion.Actualizar('Actualizar una valoracion',1,'Decepcionado',
'Muy decepcionado',TRUE);
PCK_Valoracion.Eliminar('Eliminar una valoracion',1,TRUE);
END;
/
--PRUEBAS LINEAPEDIDO----------------------------------------
------------------------------------------------------------
BEGIN
PCK_LineaPedido.Inicializar;
PCK_LineaPedido.Insertar('Insertar una linea pedido
correctamente',3,1,1,TRUE);
PCK_LineaPedido.Insertar('Insertar una linea pedido
correctamente',2,2,3,TRUE);
PCK_LineaPedido.Insertar('Insertar una linea pedido
correctamente',2,2,1,TRUE);
PCK_LineaPedido.Insertar('Insertar una linea pedido con mas productos que
stock',8,16,1,FALSE);
PCK_LineaPedido.Insertar('Insertar una linea pedido con un producto
incorrecto',1,555,1,FALSE);
PCK_LineaPedido.Insertar('Insertar una linea pedido sin pedido
asociado',2,1,null,FALSE);
PCK_LineaPedido.Actualizar('Actualizar una linea pedido',1, 1,TRUE);
PCK_LineaPedido.Eliminar('Eliminar un linea pedido',2,TRUE);
END;
/
```