PROYECTO DE IISSI
PRIMER CUATRIMESTRE

Entregable 3

Francisco Javier Abreu Vázquez

Santiago Borge Sánchez

Ángel Manuel Calzado Llamas

Ángel María Mármol Fernández

ÁRIDOS
SOFTWARE DEVELOPMENT

ÍNDICE

INTRODUCCIÓN AL PROBLEMA

Este software va dirigido a una empresa a nivel local, que se dedica al **porte y recogida de materiales de construcción** (áridos, escombros, portes especiales, etc), los cuales recoge en el polvero, que es el proveedor. La empresa, que recibe el nombre de **"Hermanos Abreu"**, consta de dos trabajadores y realiza estos transportes en Cartaya y municipios adyacentes en la provincia de Huelva. Para llevar a cabo esta tarea se usan **cubas y contenedores** que son transportadas por dos camiones, uno de ellos camión-grúa, lo que permite realizar algunos transportes más específicos.

Ambos trabajadores, **Francisco José Abreu y Manuel Abreu**, se encargan equitativamente de la dirección, aunque la empresa está a nombre del primero. Es por esto por lo que Francisco José se dedica a los temas burocráticos (facturas, vales, pago de clientes, etc.).

Actualmente la empresa usa un sistema de facturación manual, en el que se transcribe la información de los vales (con los datos del pedido) a las **facturas a mano**, que posteriormente se entregan al cliente. Además, la persona que quiera contratar los servicios de la empresa tiene que llamar por teléfono a uno de los dos trabajadores para preguntar sobre los materiales que puede transportar, el lugar donde se puede desplazar, horarios disponibles, etc. En función de estos requisitos, el trabajador tiene que ofrecerle al cliente un **presupuesto**.

Esta organización genera una serie de problemas que queremos solucionar:

En primer lugar, el sistema de facturación actual supone un gasto de tiempo innecesario pues la información sobre el pedido y su coste tiene que transcribirse de forma manual varias veces (primero al vale, que rellena el camionero con el cliente, luego a la factura y por último a un registro de facturas).

En segundo lugar, el cliente no tiene información a priori sobre los servicios que va a contratar y el precio de los mismos. Para informarse tiene que llamar al camionero, de forma que, si dos o más personas tienen que llamar, tendrán que esperar.

Nuestro objetivo es implementar un sistema de información que **automatice el sistema de facturación**, para que la empresa no pierda tiempo en los trámites cliente-trabajador, y facilitar los datos sobre los servicios que el cliente en cuestión pueda contratar, para que puedan informarse a través del software sobre los servicios y los precios.

GLOSARIO DE TÉRMINOS

- Cuba:

Contenedor de metal que se emplea para el transporte y manejo de materiales de construcción y recogida de escombros.

- Polvero:

Punto de compra de materiales áridos.

- Porte:

Transporte de la cuba (llena o vacía, dependiendo de lo que requiera el cliente) del punto de origen al destino. También se le llama servicio.

- Saca:

Saco de plástico que contiene exactamente un metro cúbico de capacidad.

- Vale:

Recibo que se rellena en el momento de la entrega y que contiene los datos del porte (Material, dirección, fecha, firma, etc.). El camionero se lleva el original y el cliente una copia.

- Factura:

Cuenta en la que se detallan las mercancías compradas o los servicios recibidos, junto con su cantidad y su importe, y que se entrega al cliente una vez se ha pagado.

- Contenedores:

Cuba de mayor tamaño que se emplea para transportes de materiales de grandes dimensiones y de poca densidad.

- Camión:

Vehículo empleado para el transporte de las cubas y los contenedores.

- Camión-Grúa:

Además de ofrecer los mismos servicios que un camión, posibilita otras actividades mediante la grúa como el levantamiento de las sacas.

- Presupuesto:

Coste a priori de un porte. Este puede variar dependiendo de la lejanía entre el cliente y la empresa, la cantidad del porte, el tipo de servicio, etc.

- Áridos:

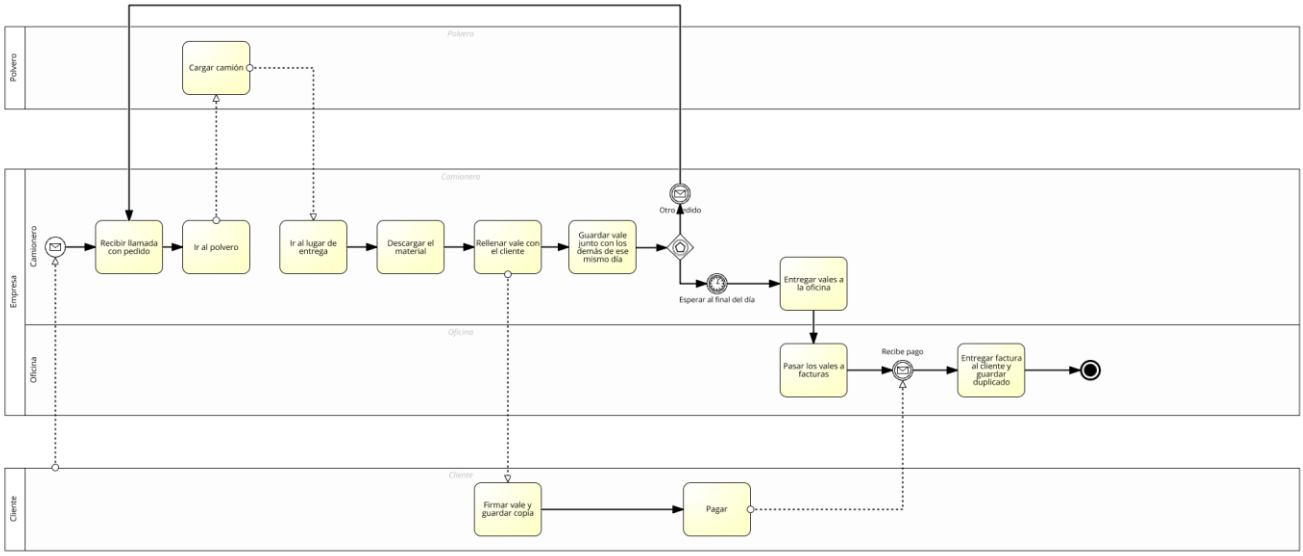
Material granulado que se utiliza como materia prima en la construcción. En nuestro caso hablamos de arena, grava, piedra caliza, etc.

-Pedido:

Servicio que el cliente solicita a la empresa.

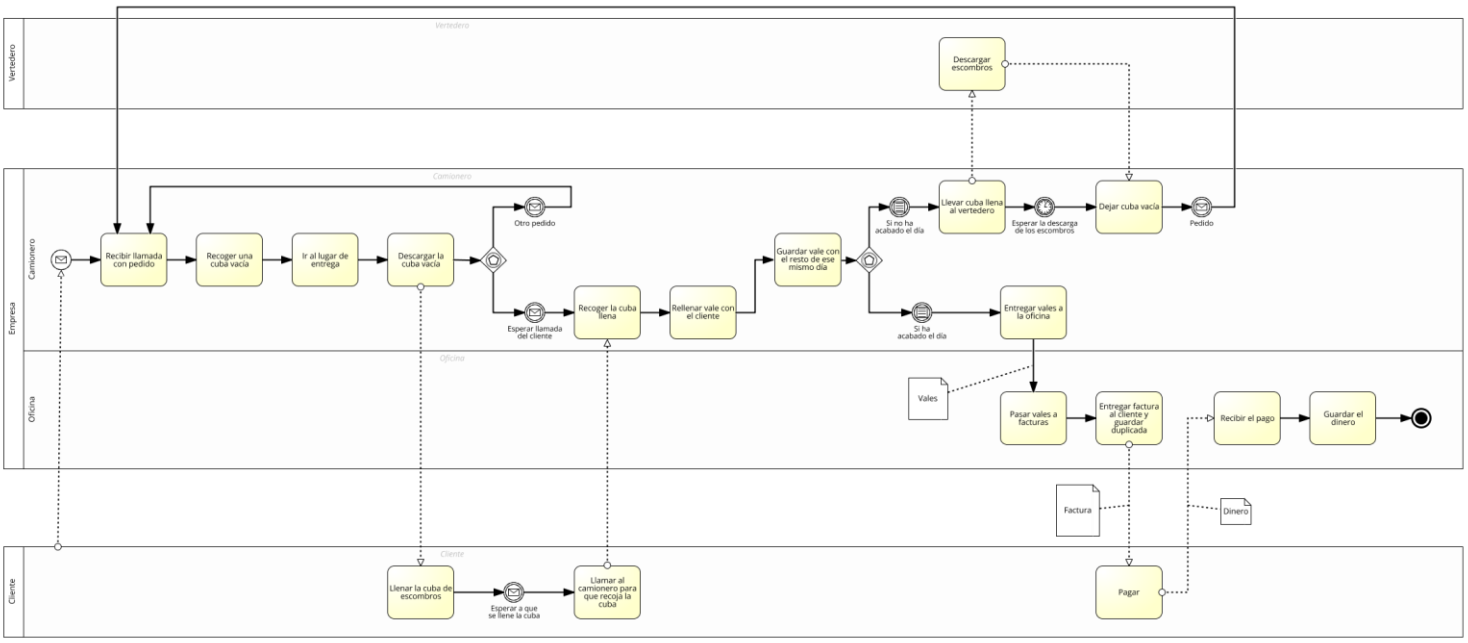
MODELO DE NEGOCIO

Porte de materiales



El cliente llama al camionero, que recibe el pedido (con el material deseado) y se dirige al polvero para recoger el material. Tras esto, se dirige al lugar de entrega, descarga el material y rellena un vale que el cliente firmará. El documento se guarda durante el resto del día, mientras el camionero sigue con otros pedidos. Al acabar la jornada, entrega los vales a la oficina, donde se transcriben a facturas que se entregan al cliente una vez ha pagado y se guarda un duplicado en la oficina. Una vez el cliente ha pagado a la oficina, se guarda el dinero.

Porte de escombros



El cliente llama al camionero para pedir una cuba vacía para escombros. El trabajador recoge una y la lleva al punto de entrega, donde el cliente se encargará de rellenarla de escombros. Entre tanto, el camionero sigue completando otros pedidos. Una vez llena, el cliente lo llamará para recogerla. Cuando recibe la llamada, recoge la cuba y rellena el vale con el cliente (que guarda junto con el resto de vales de ese día). Finalizado este proceso, lleva la cuba al vertedero, la descarga y la deja en el almacén. Tras esto, continúa completando pedidos, para, al acabar la jornada, entrega los vales a la oficina, donde se transcriben a facturas que se entregan al cliente una vez ha pagado y se guarda un duplicado en la oficina. Una vez el cliente ha pagado a la oficina, se guarda el dinero.

VISIÓN GENERAL DEL SISTEMA

El sistema que vamos a desarrollar, estará compuesto principalmente por dos partes. La primera parte, será una aplicación web donde los clientes de la empresa puedan consultar los servicios y precios que ofrece la empresa.

La segunda parte, será que dentro de aplicación el trabajador encargado de las facturas pueda acceder con un usuario privilegiado y gestionar todas las facturas de los clientes, transcribir la información de los vales a las facturas, enviar las facturas por correo electrónico, acceder a un listado de los clientes (donde puedan consultar si han abonado sus facturas, los servicios que han contratado, etc.)

Los principales usuarios de la aplicación son por parte de la empresa el encargado de gestionar las facturas, y fuera de la empresa son los clientes de ésta. A continuación, mostramos los objetivos (requisitos generales):

Presupuestos por internet
Como camionero, **quiero** que los clientes puedan consultar productos y pedir los presupuestos por Internet, **para** acelerar la gestión de los servicios

Gestión de las facturas
Como oficinista de la empresa, **quiero** poder ordenar y transcribir rápidamente las facturas, **para** ayudar al buen funcionamiento de la empresa

Consulta de productos
Como cliente de la empresa, **quiero** consultar los productos y pedir presupuesto por Internet, **para** evitar llamar cuando quiero conocer los servicios

Balance de la empresa
Como propietario de la empresa, **quiero** tener datos sobre la contabilidad, **para** tener una visión general de la trayectoria económica de la empresa.

CATÁLOGO DE REQUISITOS

Requisitos de información

RI01-Información sobre precio de los servicios
Como propietario de la empresa **quiero** tener información sobre el precio de los servicios que pide el cliente en función de: -el tipo de servicio -la cantidad -la distancia -tamaño de la cuba -horas de grúa **para** que los clientes puedan consultar un presupuesto antes de contratarlo por teléfono.

RI02-Información de los servicios
Como cliente de la empresa, **quiero** disponer de información sobre los servicios de la empresa: -todos los tipos de servicios que ofrece la empresa y sus precios **para** tener información antes de contratar los servicios.

RI03-Información de los clientes de la empresa
Como oficinista de la empresa, **quiero** tener un registro de los clientes con: -nombre del cliente -dirección del cliente -número de teléfono del cliente -correo electrónico del cliente **para** contactar con los clientes.

RI04-Registro de Pedidos
Como oficinista de la empresa, **quiero** disponer de la información sobre los pedidos de cada cliente: - cliente - precio - dirección - tipo de pedido - fecha del pedido -empleado que lo realiza **para** poder transcribir la información a la factura.

RI05-Creación de facturas
Como oficinista de la empresa, **quiero** la siguiente información en la factura: -fecha de emisión -pedidos que incluye -precio total sin IVA -precio total con IVA **para** enviarla posteriormente al cliente.

RI06-Contabilidad
Como oficinista de la empresa, **quiero** tener información sobre los gastos fijos (telefonía, autónomos, sueldo de empleados, impuestos) y gastos variables (gasolina, mantenimiento de los camiones) e ingresos **para** mantener una visión general de los fondos de la empresa.

Requisitos funcionales

RF01-Listado de Mis Pedidos
Como cliente registrado de la empresa **quiero** disponer de un listado de todos mis pedidos **para** revisar la validez del pedido.

RF02-Listado de pedidos por clientes
Como oficinista de la empresa **quiero** disponer del listado de los pedidos de cada cliente **para** comprobar el historial de pedidos de cada cliente.

RF03-Listado de pedidos por pagar
Como oficinista de la empresa **quiero** disponer de un listado de todos los pedidos que quedan por pagar **para** controlar los pagos de los clientes.

RF04-Balance
Como oficinista de la empresa **quiero** obtener un balance anual económico de la empresa (diferencia entre los ingresos y los gastos) **para** estar al tanto de la situación económica.

RF05-Generar Factura
Como oficinista de la empresa **quiero** que cuando pase el vale al sistema de información se genere una factura con los datos **para** obtener la factura.

RF06-Envío de factura
Como oficinista de la empresa **quiero** que se envía la factura por correo electrónico una vez se ha pagado el importe **para** agilizar la gestión de los pedidos.

RF07-Generar Presupuesto
Como propietario de la empresa **quiero** crear un presupuesto a partir de los datos del servicio a contratar por el cliente **para** que los clientes puedan tener un presupuesto sin tener que llamar al camionero.

RF08-Servicio especial
Como propietario de la empresa **quiero** que cuando un cliente solicite un servicio especial (transporte lejano de algún objeto pesado) se le avise que tiene que contactar con la empresa por teléfono **para** saber si es posible realizar el pedido.

RF09-Precio cuba de escombros
Como propietario de la empresa **quiero** que el precio de la cuba de escombros se calcule sólo a partir del dato distancia **porque** la cantidad es irrelevante y el tamaño de la cuba siempre es el pequeño.

RF10-Precio cuba de poda y estiércol
Como propietario de la empresa **quiero** que el precio de la cuba de poda y de estiércol se calcule sólo a partir del tamaño de la cuba y la distancia **porque** esos son los datos que influyen en el precio.

RF11-Precio cuba de áridos y de leña
Como propietario de la empresa **quiero** que los precios de la cuba de áridos y de leña se calculen en función de la cantidad y distancia **porque** esos son los datos que influyen en el precio.

RF12-Precio de servicio de grúa
Como propietario de la empresa **quiero** que el precio del servicio de la grúa se calcule a partir de las horas que lo vaya a solicitar **porque** así lo establece la empresa.

RF13-Primer pedido
Como oficinista de la empresa **quiero** que cuando se introduzca en el sistema el pedido de un cliente nuevo se registre en la base de datos **para** que el oficinista pueda tener un registro con todos los clientes

RF14-Listado de pedidos por empleados
Como oficinista de la empresa **quiero** disponer de un listado de todos los pedidos realizados por cada empleado **para** que el oficinista pueda saber cuántos pedidos ha realizado cada empleado

RF15-Listado de pedidos entre dos fechas
Como oficinista de la empresa **quiero** disponer de un listado de pedidos entre dos fechas cualesquiera **para** que el oficinista pueda ver los pedidos de un periodo concreto

RF16-Listado de los 5 clientes que deben más pedidos
Como oficinista de la empresa **quiero** disponer de una lista con los 5 clientes que deban más pedidos a la empresa **para** que el oficinista pueda ver qué clientes son los que más dinero deben

Requisitos no funcionales

RNF01-Control de gestión de facturas
Como propietario de la empresa **quiero** que solo puedan gestionar los pedidos y las facturas los empleados de mi organización **para** garantizar la seguridad de los datos.

RNF02-Claridad de los precios
Como cliente de la empresa **quiero** que la información de los precios sea lo más clara y concisa posible **para** evitar llamar al camionero solo para hacer una consulta.

Reglas de negocio

RN01-Distancia máxima
Como propietario de la empresa **quiero** que el sistema avise a los clientes de que a cierta distancia no se realizan los servicios (35 km) **para** no dar un presupuesto de un trabajo que no va a poder ser realizado.

RN02-Cantidad máxima de áridos en m^3
Como propietario de la empresa **quiero** que se establezca un límite de capacidad en cada transporte de áridos ($5m^3$) **porque** los áridos sólo se pueden transportar en las cubas de 3 y 5 (m^3).

RN03-Peso máximo de leña
Como propietario de la empresa **quiero** que se establezca un límite de peso en cada trasporte de leña (4500 Kg) **para** que no exceda de la cuba mediana.

PRUEBAS DE ACEPTACIÓN

RN01

Un cliente entra en el sistema para consultar un presupuesto del pedido que desea realizar. Este quiere 4,5m^3 de arena a una distancia de 45km, estos datos los introduce y el sistema devuelve un mensaje de error en el que dice que no se puede realizar el pedido debido a que supera la distancia máxima en la que la empresa trabaja.

RN02

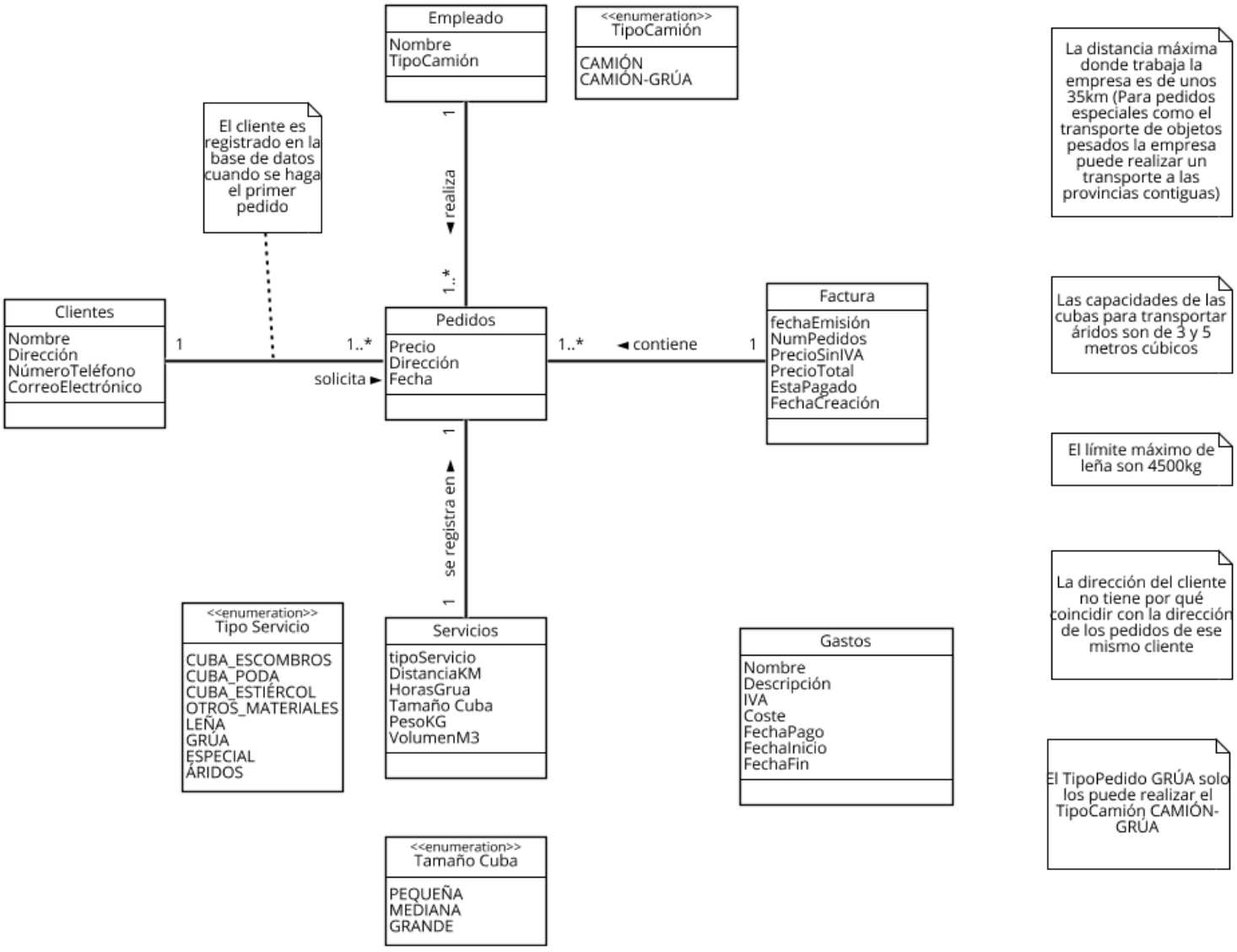
Un cliente entra en el sistema para consultar un presupuesto del pedido que desea realizar. Este quiere 6m^3 de grava a una distancia de 15km, estos datos los introduce y el sistema devuelve un mensaje de error en el que dice que no se puede realizar el pedido debido a que supera la capacidad máxima de las cubas que pueden transportar áridos.

RN03

Un cliente entra en el sistema para consultar un presupuesto del pedido que desea realizar. Este quiere 5000kg de leña a una distancia de 25km, estos datos los introduce y el sistema devuelve un mensaje de error en el que dice que no se puede realizar el pedido debido a que supera el límite de peso de las cubas.

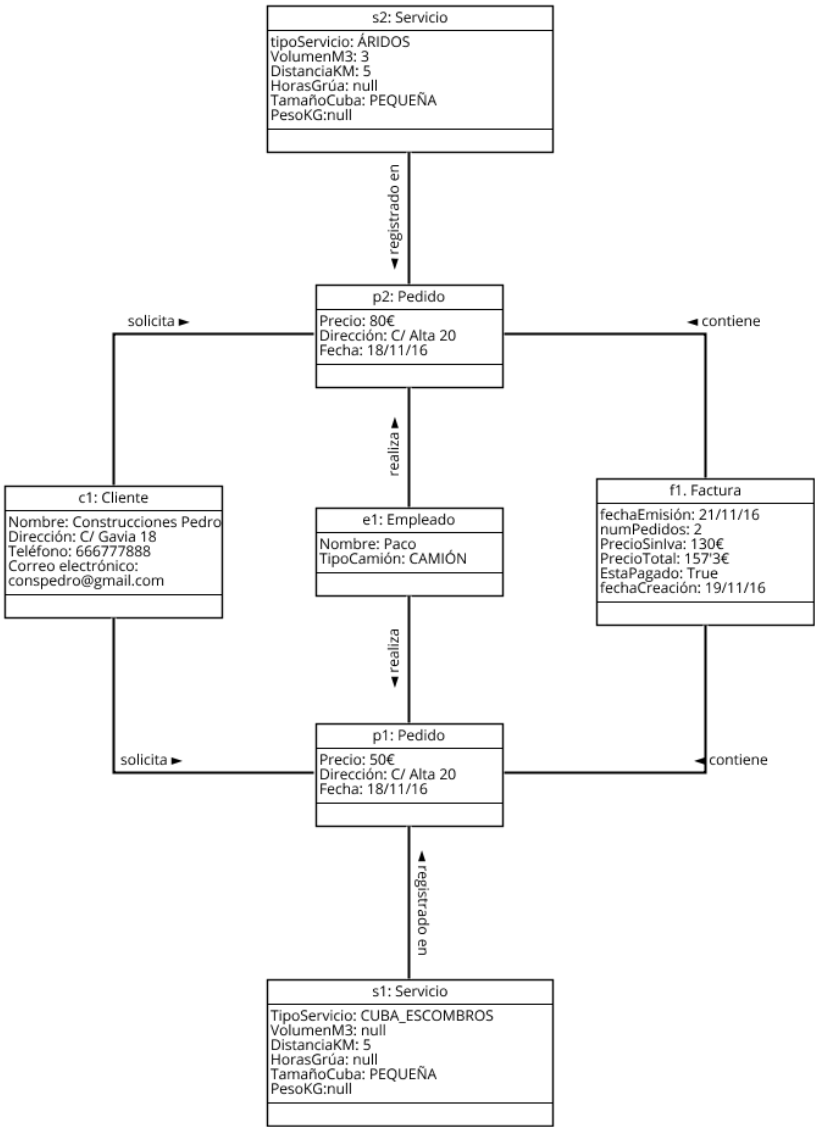
MODELADO CONCEPTUAL

DIAGRAMA UML



ESCENARIOS DE PRUEBA

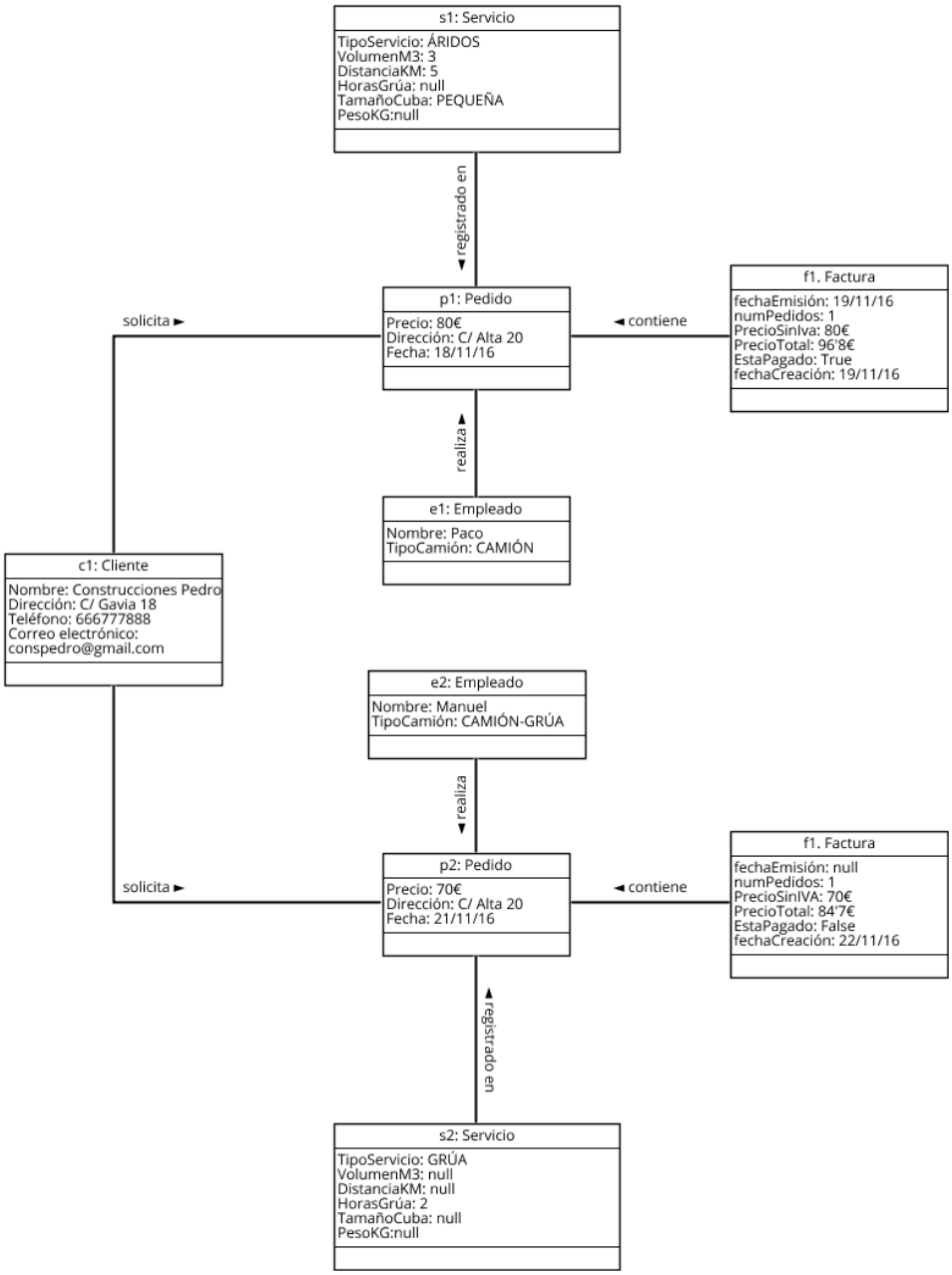
-Escenario de prueba 1



En este escenario de prueba, el cliente Construcciones Pedro realiza un pedido de una cuba de escombros (con lo cual es una cuba pequeña) a una distancia de unos 5 kilómetros, el precio de este pedido es de 50 euros y es realizado por el empleado Paco con el camión. Además, ese mismo día realiza otro pedido de 3m^3 áridos (cuba pequeña) a la misma dirección, el coste de este segundo pedido es de 80 euros y lo realiza el mismo empleado.

Los dos pedidos son registrados en la misma factura, y tres días después el cliente abona el total de los dos pedidos (157,3€), en ese momento se le envía la factura por correo.

-Escenario de prueba 2



En este escenario se prueba, el cliente Construcciones Pedro realiza un pedido de 3m^3 de áridos (cuba pequeña) a una distancia de unos kilómetros con un precio de 80€ y realizado por el empleado Paco con el camión. Al día siguiente cliente abona el total de este pedido, por lo que se le envía una factura que solo contiene este pedido. Varios días más tarde el mismo cliente solicita el servicio de la grúa durante dos horas con un precio de 70 euros y es realizado por el empleado Manuel con el camión-grúa. El importe de este pedido todavía no ha sido abonado por lo que la factura queda abierta y pendiente de ser enviada de incluir otros pedidos del mismo cliente.

-Escenario de prueba 3

En este escenario de prueba, se guardan un gasto de telefonía móvil con la compañía telefónica, dicho gasto tiene un coste de 120€ con un IVA del 21%. Este gasto corresponde al periodo comprendido entre 1/6/2016 y 31/6/2016 y fue pagado el 5/7/2016.

MATRIZ DE TRAZABILIDAD

	RI 01	RI 02	RI 03	RI 04	RI 05	RI 06	RF 01	RF 02	RF 03	RF 04	RF 05	RF 06	RF 07	RF 08	RF 09	RF 10	RF 11	RF 12	RF 13	RF 14	RF 15	RF 16	RNF 01	RNF 02	RN 01	RN 02	RN 03
Cliente			■					■						■					■			■					
Servicio	■	■											■	■	■	■	■	■						■	■	■	■
Pedido				■			■	■	■										■	■	■	■					
Empleado				■																■			■				
Factura					■				■	■	■	■											■				
Gastos						■				■																	
solicita	■	■		■			■	■											■								
registrado en				■																							
realiza				■																							
contiene					■						■																

MODELO RELACIONAL EN 3FN

Relaciones	PK	FK	AK
Clientes{ OID_C, Nombre, Dirección, NúmeroTeléfono, CorreoElectrónico}	OID_C		CorreoElectrónico
Empleados{ OID_E, NombreE, TipoCamion}	OID_E		
Servicios{ OID_S, tipoServicio, DistanciaKM, HorasGrua, Tamaño Cuba, PesoKG, VolumenM3}	OID_S		
Facturas{ OID_F, fechaEmisión, NumPedidos, PrecioTotal, PrecioIVA, EstaPagado, FechaCreación}	OID_F		
Gastos{ OID_G, Nombre, Descripción, IVA, Coste, FechaPago, FechaInicio, FechaFin}	OID_G		
Pedidos{ OID_P, Precio, Dirección, Fecha}	OID_P	OID_C, OID_E, OID_S, OID_F	

MODELO TECNOLÓGICO

```
------------BORRADO DE SECUENCIAS-------------
DROP SEQUENCE sec_clientes;
DROP SEQUENCE sec_empleados;
DROP SEQUENCE sec_servicios;
DROP SEQUENCE sec_facturas;
DROP SEQUENCE sec_pedidos;
DROP SEQUENCE sec_gastos;

------------BORRADO DE TABLAS--------------
DROP TABLE Pedidos;
DROP TABLE Clientes;
DROP TABLE Empleados;
DROP TABLE Servicios;
DROP TABLE Facturas;
DROP TABLE Gastos;


------------CREACION DE TABLAS--------------
CREATE TABLE Clientes(
  OID_C INTEGER NOT NULL,
  Nombre VARCHAR2(50) NOT NULL,
  Direccion VARCHAR2(50),
  NumeroTelefono VARCHAR2(9) NOT NULL,
  CorreoElectronico VARCHAR2(60) NOT NULL,
  PRIMARY KEY(OID_C),
  UNIQUE(CorreoElectronico)
  );

CREATE TABLE Empleados(
  OID_E INTEGER NOT NULL,
  NombreE VARCHAR2(50) NOT NULL,
  TipoCamion VARCHAR2(20) NOT NULL, CONSTRAINT "TIPOCAMION_CHK1" CHECK
(TipoCamion IN('Camion','Camion_Grua')),
  PRIMARY KEY(OID_E)
  );
```

```
CREATE TABLE Servicios(
  OID_S INTEGER NOT NULL,
  TipoServicio VARCHAR2(25) NOT NULL, CONSTRAINT "TIPOSERVICIO_CHK2" CHECK
(TipoServicio
IN('CUBA_ESCOMBROS','CUBA_PODA','CUBA_ESTIERCOL','OTRO_MATERIALES','LEÑA','GRUA','
ESPECIAL','ARIDOS')),
  DistanciaKM INTEGER, CONSTRAINT "DISTANCIA_CHK5" CHECK (DistanciaKM<=35),
  HorasGrua INTEGER,
  TamañoCuba VARCHAR2(10), CONSTRAINT "TAMAÑOCUBA_CHK3" CHECK (TamañoCuba
IN('PEQUEÑA','MEDIANA','GRANDE')),
  PesoKG INTEGER, CONSTRAINT "PesoKG" CHECK (PesoKG<4500),
  VolumenM3 INTEGER, CONSTRAINT "VOLUMENM3_CHK7" CHECK (VolumenM3<=5),
  PRIMARY KEY(OID_S)
  );

CREATE TABLE Facturas(
  OID_F INTEGER NOT NULL,
  FechaEmision DATE, CONSTRAINT "FECHAEMISON_CHK6" CHECK (FechaEmision >
FechaCreacion),
  NumPedidos INTEGER NOT NULL,
  PrecioSinIva NUMBER(6,2) NOT NULL,
  PrecioTotal NUMBER (6,2) NOT NULL,
  EstaPagado CHAR(2) NOT NULL, CONSTRAINT "ESTAPAGADO_CHK4"CHECK (EstaPagado
IN('SI','NO')),
  FechaCreacion DATE DEFAULT SYSDATE,
  PRIMARY KEY(OID_F)
  );

CREATE TABLE Gastos(
  OID_G INTEGER NOT NULL,
  Nombre VARCHAR2(50) NOT NULL,
  Descripcion VARCHAR2(70) NOT NULL,
  Coste NUMBER(6,2) NOT NULL,
  IVA NUMBER(6,2) NOT NULL,
  FechaPago DATE NOT NULL,
  FechaInicio DATE NOT NULL,
  FechaFin DATE NOT NULL, CONSTRAINT "FECHAFIN_CHK8" CHECK (FechaInicio<=FechaFin),
  PRIMARY KEY(OID_G)
  );
```

```
CREATE TABLE Pedidos(
  OID_P INTEGER NOT NULL,
  Precio NUMBER(6,2) NOT NULL,
  Direccion VARCHAR(50) NOT NULL,
  Fecha DATE NOT NULL,
  OID_C INTEGER NOT NULL,
  OID_E INTEGER NOT NULL,
  OID_S INTEGER NOT NULL,
  OID_F INTEGER NOT NULL,

  PRIMARY KEY(OID_P),
  FOREIGN KEY(OID_C) REFERENCES Clientes,
  FOREIGN KEY(OID_E) REFERENCES Empleados,
  FOREIGN KEY(OID_S) REFERENCES Servicios,
  FOREIGN KEY(OID_F) REFERENCES Facturas
  );



----------CREACION DE SECUENCIAS----------
CREATE SEQUENCE sec_clientes;
CREATE OR REPLACE TRIGGER crea_oid_c BEFORE INSERT ON Clientes
FOR EACH ROW
BEGIN
        SELECT sec_clientes.NEXTVAL INTO :NEW.OID_C FROM DUAL;
END;
/
CREATE SEQUENCE sec_empleados;
CREATE OR REPLACE TRIGGER crea_oid_e BEFORE INSERT ON Empleados
FOR EACH ROW
BEGIN
        SELECT sec_empleados.NEXTVAL INTO :NEW.OID_E FROM DUAL;
END;
/
CREATE SEQUENCE sec_servicios;
CREATE OR REPLACE TRIGGER crea_oid_s BEFORE INSERT ON Servicios
FOR EACH ROW
BEGIN
        SELECT sec_servicios.NEXTVAL INTO :NEW.OID_S FROM DUAL;
END;
/
CREATE SEQUENCE sec_facturas;
CREATE OR REPLACE TRIGGER crea_oid_f BEFORE INSERT ON Facturas
FOR EACH ROW
BEGIN
        SELECT sec_facturas.NEXTVAL INTO :NEW.OID_F FROM DUAL;
END;
/
```

```
CREATE SEQUENCE sec_gastos;
CREATE OR REPLACE TRIGGER crea_oid_g BEFORE INSERT ON Gastos
FOR EACH ROW
BEGIN
        SELECT sec_gastos.NEXTVAL INTO :NEW.OID_G FROM DUAL;
END;
/
CREATE SEQUENCE sec_pedidos;
CREATE OR REPLACE TRIGGER crea_oid_p BEFORE INSERT ON Pedidos
FOR EACH ROW
BEGIN
        SELECT sec_pedidos.NEXTVAL INTO :NEW.OID_P FROM DUAL;
END;
/


------------TRIGGERS------------
--trigger IVA gastos--

CREATE OR REPLACE TRIGGER trigger_IVA_gastos
BEFORE INSERT OR UPDATE OF COSTE, IVA ON GASTOS
FOR EACH ROW
BEGIN
   :NEW.IVA := :New.Coste-(:NEW.Coste/1.21);
END;
/

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




--trigger IVA facturas--

CREATE OR REPLACE TRIGGER trigger_IVA_facturas
BEFORE INSERT OR UPDATE OF PRECIOTOTAL, PRECIOSINIVA ON FACTURAS
FOR EACH ROW
BEGIN
        :NEW.PrecioTotal := :NEW.PrecioSinIva*1.21;
END;
/

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

```
--trigger fecha emision--

CREATE OR REPLACE TRIGGER trigger_fecha_emision
BEFORE UPDATE OR INSERT ON FACTURAS
FOR EACH ROW
BEGIN
        IF (:NEW.EstaPagado = 'NO' AND :NEW.FechaEmision IS NOT null)
        THEN raise_application_error(-20600,'La fecha de emision no puede ser distinto de null
si la factura no esta pagada');
        END IF;
END;
/


--trigger fecha emision 2--
CREATE OR REPLACE TRIGGER trigger_fecha_emision_2
BEFORE UPDATE OF ESTAPAGADO OR INSERT ON FACTURAS
FOR EACH ROW
BEGIN
        IF :NEW.EstaPagado = 'SI'

        THEN :NEW.FechaEmision := SYSDATE;

        END IF;
END;
/

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

--trigger asignacion factura--

CREATE OR REPLACE TRIGGER asignacion_factura
BEFORE INSERT ON Pedidos
FOR EACH ROW
DECLARE
CONTADOR integer;
OIDFACTURA integer;
BEGIN
SELECT COUNT(*) into CONTADOR FROM (SELECT * FROM facturas F NATURAL JOIN pedidos
P WHERE (P.OID_C=:NEW.OID_C AND F.estapagado='NO') ORDER BY F.FECHACREACION
DESC) WHERE ROWNUM =1;
IF CONTADOR =0

   THEN
   INSERT INTO FACTURAS(NUMPEDIDOS, PRECIOSINIVA, ESTAPAGADO, FECHACREACION)
VALUES (1, :NEW.PRECIO, 'NO', SYSDATE);
   :NEW.OID_F := sec_facturas.CURRVAL;
   ELSE
   SELECT oid_f into OIDFACTURA FROM (SELECT OID_F FROM facturas F NATURAL JOIN
pedidos P WHERE (P.OID_C=:NEW.OID_C AND F.estapagado='NO') ORDER BY
F.FECHACREACION DESC) WHERE ROWNUM =1;
   :NEW.OID_F := OIDFACTURA;
```

```
   UPDATE FACTURAS F SET F.PRECIOSINIVA = (F.preciosiniva + :NEW.PRECIO),
F.NUMPEDIDOS = (F.NUMPEDIDOS + 1)   WHERE F.OID_F = oid_f;


         END IF;
END;
/
-------------------------------------------------------------------------------------------------------------------
--
-- Trigger borrar_pedido--
-- Actualiza los datos de factura y servicios al borrar un pedido

CREATE OR REPLACE TRIGGER borrar_pedido
AFTER DELETE ON Pedidos
FOR EACH ROW
DECLARE
NUMERO INTEGER;

BEGIN
UPDATE FACTURAS F SET F.NUMPEDIDOS = F.NUMPEDIDOS -1, F.PRECIOSINIVA =
F.PRECIOSINIVA - :OLD.PRECIO WHERE F.OID_F = :OLD.OID_F RETURNING F.NUMPEDIDOS
INTO NUMERO;

IF NUMERO <= 0
   THEN DELETE FACTURAS F WHERE F.OID_F = :OLD.OID_F;
   END IF;

DELETE SERVICIOS S WHERE S.OID_S = :OLD.OID_S;

END;
/
--SCRIPT CONSULTAS

------------------------------RF03 (Lista de pedidos que están por pagar)

SELECT * FROM Facturas NATURAL JOIN pedidos WHERE EstaPagado='NO' ORDER BY
pedidos.oid_c;
```

```
------------------------------RF17 (Los cinco clientes que deban más dinero a la empresa)

DECLARE

CURSOR cur_morosos IS
  SELECT NOMBRE, PrecioSinIva FROM CLIENTES C, PEDIDOS P, FACTURAS F WHERE
F.ESTAPAGADO='NO' and (P.OID_C=C.OID_C AND P.OID_F = F.OID_F) ORDER BY
F.PrecioSinIva DESC;

BEGIN
    FOR fila IN cur_morosos LOOP
    EXIT WHEN cur_morosos%ROWCOUNT >5;
    DBMS_OUTPUT.PUT_LINE(fila.NOMBRE);
    END LOOP;
END;
/



----------------------------CONSULTA QUE DEVUELVE LISTA DE PEDIDOS DE UNA FACTURA

Select Nombre, TipoServicio, DistanciaKM,  PesoKG, VolumenM3, TamañoCuba, Precio
HorasGrua FROM Pedidos P, Servicios S, Facturas F, Clientes C WHERE (P.OID_C=C.OID_C and
P.OID_F=F.OID_F) ORDER BY P.Fecha;



-------------------------------------------------------------------------------------------------------------------
-----------
------------------------PROCEDIMIENTOS-----------------------------------------------------------------------
--------------------
-------------------------------------------------------------------------------------------------------------------
-----------

--Procedimiento lista de pedidos de un cliente (RF01 & RF02)

create or replace PROCEDURE pedidos_por_cliente (w_OID_C IN CLIENTES.OID_C%TYPE)
    IS

CURSOR cur_ped_cliente IS SELECT Nombre, TipoServicio, Fecha FROM Pedidos P, Servicios S,
Clientes C WHERE (P.OID_C=W_OID_C and C.OID_C = W_OID_C AND P.OID_S = S.OID_S)
ORDER BY Nombre;
BEGIN

FOR fila IN cur_ped_cliente LOOP
EXIT WHEN cur_ped_cliente%notfound;
DBMS_OUTPUT.PUT_LINE(fila.nombre||' '||fila.tiposervicio||' '||fila.fecha);
END LOOP;


END;
/
```

```
--Procedimiento pedidos por empleados (RF15)
create or replace PROCEDURE pedidos_por_empleado (w_OID_E IN
EMPLEADOS.OID_E%TYPE)
        IS

CURSOR cur_ped_empleado IS SELECT NombreE, TipoServicio, Fecha FROM Pedidos P,
Servicios S, Empleados E WHERE (P.OID_E=W_OID_E and E.OID_E = W_OID_E AND P.OID_S
= S.OID_S) ORDER BY NombreE;

BEGIN

FOR fila IN cur_ped_empleado LOOP
EXIT WHEN cur_ped_empleado%notfound;
DBMS_OUTPUT.PUT_LINE(fila.nombreE||' '||fila.tiposervicio||' '||fila.fecha);
END LOOP;

END;
/
--Procedimiento pedidos entre dos fechas
create or replace PROCEDURE pedidos_por_fecha (w_fecha_inicio IN Pedidos.fecha%TYPE,
w_fecha_fin IN Pedidos.fecha%TYPE)
        IS

CURSOR cur_ped_fechas IS SELECT Nombre, TipoServicio, Fecha FROM Pedidos P, Servicios S,
Clientes C WHERE (P.fecha BETWEEN (w_fecha_inicio) and (w_fecha_fin));

BEGIN

FOR fila IN cur_ped_fechas LOOP
EXIT WHEN cur_ped_fechas%notfound;
DBMS_OUTPUT.PUT_LINE(fila.nombre||' '||fila.tiposervicio||' '||fila.fecha);
END LOOP;

END;
```

```
--------------------------------------------------------------------------------------------------------------
----------------------FUNCIONES-----------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
---------------------------------FUNCION CALCULAR PRECIO-------------------------
CREATE OR REPLACE FUNCTION calc_precio
  (tipo_servicio IN Servicios.TipoServicio%TYPE, distancia_km IN Servicios.DistanciaKM%TYPE,
peso_kg IN Servicios.PesoKG%TYPE,
  volumen_m3 IN Servicios.VolumenM3%TYPE, horas_grua IN Servicios.HorasGrua%TYPE,
cuba IN Servicios.TamañoCuba%TYPE)
  RETURN NUMBER IS precio PEDIDOS.PRECIO%TYPE;

BEGIN

RETURN CASE tipo_servicio
    WHEN 'GRUA'
        THEN (horas_grua*35)

    WHEN 'CUBA_ESCOMBROS'
        THEN
            (CASE
                WHEN (distancia_km<16)
                    THEN 50
                ELSE 60
            END)

    WHEN 'CUBA_PODA'
        THEN
            (CASE cuba
                WHEN 'PEQUEÑA'
                    THEN 50
                WHEN 'MEDIANA'
                    THEN 80
                ELSE 100
            END)

            WHEN 'CUBA_ESTIERCOL'
                    THEN
                            (CASE cuba
                                    WHEN 'PEQUEÑA'
                                            THEN 60

                                    ELSE 80
                            END)

            WHEN 'ARIDOS'
                    THEN
                            (CASE
                                    WHEN (distancia_km<16)
                                            THEN 50
                                    ELSE 60
                            END)

            WHEN 'LEÑA'
```

```
                THEN (peso_kg*0.12)

        WHEN 'OTRO_MATERIALES'
                THEN
                        (CASE
                                WHEN (distancia_km<16)
                                        THEN 50
                                ELSE 60
                        END)
   ELSE 0
   END;

END calc_precio;
/
```

```
---------------------------------------FUNCION BALANCE-------------------------------

CREATE OR REPLACE FUNCTION balance
        (fechaInicial DATE, fechaFinal DATE)

RETURN NUMBER IS balance NUMBER(6,2);

SumaIngresos NUMBER(6,2);
SumaGastos NUMBER(6,2);

BEGIN

SELECT SUM(PrecioTotal) INTO SumaIngresos FROM facturas F WHERE (F.EstaPagado='SI' and
F.FechaEmision between (FechaInicial) and (FechaFinal));

SELECT SUM(Coste) INTO SumaGastos FROM gastos G WHERE (G.FechaPago between
(FechaInicial) and (FechaFinal));

RETURN SumaIngresos-SumaGastos;

END;

/
```

```
----------Especificacion del paquete PRUEBAS_GASTOS----------
create or replace
PACKAGE PRUEBAS_GASTOS AS
        PROCEDURE inicializar;
        PROCEDURE insertar
        (nombre_prueba VARCHAR2, W_Nombre VARCHAR2, W_Descripcion VARCHAR2,
W_Coste NUMBER, W_FechaPago DATE, W_FechaInicio DATE, W_FechaFin DATE,
salidaEsperada BOOLEAN);
        PROCEDURE actualizar
        (nombre_prueba VARCHAR2, W_OID_G INTEGER, W_Nombre VARCHAR2,
W_Descripcion VARCHAR2, W_Coste NUMBER, W_FechaPago DATE, W_FechaInicio DATE,
W_FechaFin DATE, salidaEsperada BOOLEAN);
        PROCEDURE eliminar
        (nombre_prueba VARCHAR2, W_OID_G INTEGER, salidaEsperada BOOLEAN);

END PRUEBAS_GASTOS;
/

----------CUERPO DEL PAQUETE PRUEBAS_GASTOS----------
create or replace
PACKAGE BODY PRUEBAS_GASTOS AS
        --INICIALIZACION
        PROCEDURE inicializar AS
        BEGIN
        --Borrar contenido de la tabla--
                DELETE FROM GASTOS;
        END inicializar;

        --PRUEBA PARA LA INSERCION DE GASTOS--
        PROCEDURE insertar (nombre_prueba VARCHAR2, W_Nombre VARCHAR2,
W_Descripcion VARCHAR2, W_Coste NUMBER, W_FechaPago DATE, W_FechaInicio DATE,
W_FechaFin DATE, salidaEsperada BOOLEAN) AS
                salida BOOLEAN := true;
                gasto gastos%ROWTYPE;
                W_OID_G INTEGER;
        BEGIN
                --Insertar gasto--
                INSERT INTO gastos(NOMBRE, DESCRIPCION, COSTE, FECHAPAGO,
FECHAINICIO, FECHAFIN) VALUES(W_Nombre,W_Descripcion, W_Coste, W_FechaPago,
W_FechaInicio, W_FechaFin);
                --Seleccionar gasto y comprobar que los datos se insertan correctamente--
                W_OID_G:= sec_gastos.currval;
                SELECT * INTO gasto FROM gastos WHERE OID_G = W_OID_G;
                IF(GASTO.NOMBRE<>W_Nombre and GASTO.Descripcion<>W_Descripcion
and GASTO.Coste<>W_Coste and GASTO.FechaPago<>W_FechaPago and
GASTO.FechaInicio<>W_FechaInicio and GASTO.FechaFin<>W_FechaFin) THEN
                        salida:=false;
                END IF;
                COMMIT WORK;
```

```
		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

		EXCEPTION
		WHEN OTHERS THEN
			DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
			ROLLBACK;
	END insertar;

	--PRUEBA PARA LA ACTUALIZACION DE GASTOS--
	PROCEDURE actualizar (nombre_prueba VARCHAR2, W_OID_G INTEGER, W_Nombre
VARCHAR2, W_Descripcion VARCHAR2, W_Coste NUMBER, W_FechaPago DATE, W_FechaInicio
DATE, W_FechaFin DATE, salidaEsperada BOOLEAN) AS
		salida BOOLEAN := true;
		gasto gastos%ROWTYPE;
	BEGIN
		--Actualizar empleado--
		UPDATE gastos SET Nombre=W_Nombre, Descripcion=W_Descripcion,
Coste=W_coste, FechaPago=W_FechaPago, FechaInicio=W_FechaInicio,
FechaFin=W_FechaFin WHERE OID_G=W_OID_G;

		--Seleccionar gasto y comprobar que los campos se actualizaron correctamente-
-
		SELECT * INTO gasto FROM GASTOS WHERE OID_G = W_OID_G;
		IF(GASTO.NOMBRE<>W_Nombre and GASTO.Descripcion<>W_Descripcion
and GASTO.Coste<>W_Coste and GASTO.FechaPago<>W_FechaPago and
GASTO.FechaInicio<>W_FechaInicio and GASTO.FechaFin<>W_FechaFin) THEN
			salida:= false;
		END IF;
		COMMIT WORK;

		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

		EXCEPTION
		WHEN OTHERS THEN
			DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
			ROLLBACK;
	END actualizar;

	--PRUEBA PARA LA ELIMINACION DE GASTOS--
	PROCEDURE eliminar (nombre_prueba VARCHAR2, W_OID_G INTEGER, salidaEsperada
BOOLEAN) AS
	salida BOOLEAN := true;
	n_gastos INTEGER;
	BEGIN
		--Eliminar gasto--
		DELETE FROM gastos WHERE OID_G=W_OID_G;
```

```
		--Verificar que el gasto no se encuentra en la BD--
		SELECT COUNT (*) INTO n_gastos FROM gastos WHERE OID_G=W_OID_G;
		IF(n_gastos<>0) THEN
			salida:=false;
		END IF;
		COMMIT WORK;

		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

		EXCEPTION
		WHEN OTHERS THEN
			DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
			ROLLBACK;
	END eliminar;
END PRUEBAS_GASTOS;
/


--------------------PRUEBA_CLIENTES----------------------------------------------------------------
create or replace
PACKAGE PRUEBAS_CLIENTES AS

	PROCEDURE inicializar;
	PROCEDURE insertar
		(nombre_prueba VARCHAR2, W_Nombre VARCHAR2, W_Direccion VARCHAR2,
W_NumeroTelefono VARCHAR2, W_CorreoElectronico VARCHAR2, salidaEsperada BOOLEAN);
	PROCEDURE actualizar
		(nombre_prueba VARCHAR2, W_OID_C INTEGER, W_Nombre VARCHAR2,
W_Direccion VARCHAR2, W_NumeroTelefono VARCHAR2, W_CorreoElectronico VARCHAR2,
salidaEsperada BOOLEAN);
	PROCEDURE eliminar
		(nombre_prueba VARCHAR2, W_OID_C INTEGER, salidaEsperada BOOLEAN);

END PRUEBAS_CLIENTES;
/

create or replace
PACKAGE BODY PRUEBAS_CLIENTES AS
	--INICIALIZACION
	PROCEDURE inicializar AS
	BEGIN
	--Borrar contenido de la tabla--
		DELETE FROM CLIENTES;
	END inicializar;

	--PRUEBA PARA LA INSERCION DE CLIENTES--
	PROCEDURE insertar (nombre_prueba VARCHAR2, W_Nombre VARCHAR2, W_Direccion
VARCHAR2, W_NumeroTelefono VARCHAR2, W_CorreoElectronico VARCHAR2, salidaEsperada
BOOLEAN) AS
		salida BOOLEAN := true;
```

```
                cliente clientes%ROWTYPE;
                W_OID_C INTEGER;
        BEGIN
                --Insertar cliente--
                INSERT INTO clientes(NOMBRE, DIRECCION,
NUMEROTELEFONO,CORREOELECTRONICO) VALUES(W_Nombre,W_Direccion,
W_NumeroTelefono, W_CorreoElectronico);
                --Seleccionar cliente y comprobar que los datos se insertan correctamente--
                W_OID_C:= sec_clientes.currval;
                SELECT * INTO cliente FROM clientes c WHERE c.OID_C = W_OID_C;
                IF(CLIENTE.NOMBRE<>W_Nombre and CLIENTE.Direccion<>W_Direccion and
CLIENTE.NumeroTelefono<>W_NumeroTelefono and
CLIENTE.CorreoElectronico<>W_CorreoElectronico) THEN
                        salida:=false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END insertar;

--PRUEBA PARA LA ACTUALIZACION DE CLIENTE--
        PROCEDURE actualizar (nombre_prueba VARCHAR2, W_OID_C INTEGER, W_Nombre
VARCHAR2, W_Direccion VARCHAR2, W_NumeroTelefono VARCHAR2, W_CorreoElectronico
VARCHAR2, salidaEsperada BOOLEAN) AS
                salida BOOLEAN := true;
                cliente clientes%ROWTYPE;
        BEGIN
                --Actualizar cliente--
                UPDATE clientes SET Nombre=W_Nombre, Direccion=W_Direccion,
NumeroTelefono=W_NumeroTelefono, CorreoElectronico=W_CorreoElectronico WHERE
OID_C=W_OID_C;

                --Seleccionar cliente y comprobar que los campos se actualizaron
correctamente--
                SELECT * INTO cliente FROM CLIENTES c WHERE c.OID_C = W_OID_C;
                IF(CLIENTE.NOMBRE<>W_Nombre and CLIENTE.Direccion<>W_Direccion and
CLIENTE.NumeroTelefono<>W_NumeroTelefono and
CLIENTE.CorreoElectronico<>W_CorreoElectronico) THEN
                        salida:= false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));
```

```
                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END actualizar;

--PRUEBA PARA LA ELIMINACION DE CLIENTES--
        PROCEDURE eliminar (nombre_prueba VARCHAR2, W_OID_C INTEGER, salidaEsperada
BOOLEAN) AS
        salida BOOLEAN := true;
        n_clientes INTEGER;
        BEGIN
                --Eliminar cliente--
                DELETE FROM CLIENTES C WHERE C.OID_C=W_OID_C;

                --Verificar que el cliente no se encuentra en la BD--
                SELECT COUNT (*) INTO n_clientes FROM clientes WHERE OID_C=W_OID_C;
                IF(n_clientes<>0) THEN
                        salida:=false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END eliminar;
END PRUEBAS_CLIENTES;
/

--------------PRUEBA EMPLEADOS--------------------------------------------------------------------
create or replace
PACKAGE PRUEBAS_EMPLEADOS AS
        PROCEDURE inicializar;
        PROCEDURE insertar
        (nombre_prueba VARCHAR2, W_NombreE VARCHAR2, W_TipoCamion VARCHAR2,
salidaEsperada BOOLEAN);
        PROCEDURE actualizar
        (nombre_prueba VARCHAR2, W_OID_E INTEGER, W_NombreE VARCHAR2,
W_TipoCamion VARCHAR2, salidaEsperada BOOLEAN);
        PROCEDURE eliminar
        (nombre_prueba VARCHAR2, W_OID_E INTEGER, salidaEsperada BOOLEAN);

END PRUEBAS_EMPLEADOS;
/
--------------------CUERPO DEL PAQUETE PRUEBAS_EMPLEADOS--------------------
```

```
create or replace
PACKAGE BODY PRUEBAS_EMPLEADOS AS
	--INICIALIZACION
	PROCEDURE inicializar AS
	BEGIN
	--Borrar contenido de la tabla--
		DELETE FROM EMPLEADOS;
	END inicializar;

	--PRUEBA PARA LA INSERCION DE EMPLEADOS--
	PROCEDURE insertar (nombre_prueba VARCHAR2, W_NombreE VARCHAR2,
W_TipoCamion VARCHAR2, salidaEsperada BOOLEAN) AS
		salida BOOLEAN := true;
		empleado empleados%ROWTYPE;
		W_OID_E INTEGER;
	BEGIN
		--Insertar empleado--
		INSERT INTO empleados(NOMBREE, TIPOCAMION) VALUES(W_NombreE,
W_TipoCamion);
		--Seleccionar empleado y comprobar que los datos se insertan correctamente--
		W_OID_E:= sec_empleados.currval;
		SELECT * INTO empleado FROM empleados WHERE OID_E = W_OID_E;
		IF(EMPLEADO.NOMBREE<>W_NombreE and
EMPLEADO.TIPOCAMION<>W_TipoCamion) THEN
			salida:=false;
		END IF;
		COMMIT WORK;

		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

		EXCEPTION
		WHEN OTHERS THEN
			DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
			ROLLBACK;
	END insertar;

	--PRUEBA PARA LA ACTUALIZACION DE EMPLEADOS--
	PROCEDURE actualizar (nombre_prueba VARCHAR2, W_OID_E INTEGER, W_NombreE
VARCHAR2, W_TipoCamion VARCHAR2, salidaEsperada BOOLEAN) AS
		salida BOOLEAN := true;
		empleado empleados%ROWTYPE;
	BEGIN
		--Actualizar empleado--
		UPDATE empleados SET NombreE=W_NombreE, TipoCamion=W_TipoCamion
WHERE OID_E=W_OID_E;

		--Seleccionar gasto y comprobar que los campos se actualizaron correctamente-
-
		SELECT * INTO empleado FROM EMPLEADOS WHERE OID_E = W_OID_E;
```

```
                IF(EMPLEADO.NOMBREE<>W_NombreE and
EMPLEADO.TIPOCAMION<>W_TipoCamion) THEN
                        salida:= false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END actualizar;

        --PRUEBA PARA LA ELIMINACION DE EMPLEADOS--
        PROCEDURE eliminar (nombre_prueba VARCHAR2, W_OID_E INTEGER, salidaEsperada
BOOLEAN) AS
        salida BOOLEAN := true;
        n_empleados INTEGER;
        BEGIN
                --Eliminar empleado--
                DELETE FROM empleados WHERE OID_E=W_OID_E;

                --Verificar que el empleado no se encuentra en la BD--
                SELECT COUNT (*) INTO n_empleados FROM empleados WHERE
OID_E=W_OID_E;
                IF(n_empleados<>0) THEN
                        salida:=false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END eliminar;
END PRUEBAS_EMPLEADOS;
/
```

```
----------------PRUEBA_SERVICIOS-------------------------------------------------------------------------
create or replace
PACKAGE PRUEBAS_SERVICIOS AS
	PROCEDURE inicializar;
	PROCEDURE insertar
	(nombre_prueba VARCHAR2, W_TipoServicio VARCHAR2, W_DistanciaKM INTEGER,
W_HorasGrua INTEGER, W_TamañoCuba VARCHAR2, W_PesoKG INTEGER, W_VolumenM3
INTEGER, salidaEsperada BOOLEAN);
	PROCEDURE actualizar
	(nombre_prueba VARCHAR2, W_OID_S INTEGER, W_TipoServicio VARCHAR2,
W_DistanciaKM INTEGER, W_HorasGrua INTEGER, W_TamañoCuba VARCHAR2, W_PesoKG
INTEGER, W_VolumenM3 INTEGER, salidaEsperada BOOLEAN);
	PROCEDURE eliminar
	(nombre_prueba VARCHAR2, W_OID_S INTEGER, salidaEsperada BOOLEAN);

END PRUEBAS_SERVICIOS;
/
--------------------CUERPO DEL PAQUETE PRUEBAS_SERVICIOS--------------------
create or replace
PACKAGE BODY PRUEBAS_SERVICIOS AS
	--INICIALIZACION
	PROCEDURE inicializar AS
	BEGIN
	--Borrar contenido de la tabla--
		DELETE FROM SERVICIOS;
	END inicializar;

	--PRUEBA PARA LA INSERCION DE SERVICIOS--
	PROCEDURE insertar (nombre_prueba VARCHAR2, W_TipoServicio VARCHAR2,
W_DistanciaKM INTEGER, W_HorasGrua INTEGER, W_TamañoCuba VARCHAR2, W_PesoKG
INTEGER, W_VolumenM3 INTEGER, salidaEsperada BOOLEAN) AS
		salida BOOLEAN := true;
		servicio servicios%ROWTYPE;
		W_OID_S INTEGER;
	BEGIN
		--Insertar servicio--
		INSERT INTO servicios(TIPOSERVICIO,DISTANCIAKM, HORASGRUA,
TAMAÑOCUBA,PESOKG,VOLUMENM3) VALUES(W_TipoServicio, W_DistanciaKM, W_HorasGrua,
W_TamañoCuba, W_PesoKG, W_VolumenM3);
		--Seleccionar empleado y comprobar que los datos se insertan correctamente--
		W_OID_S:= sec_servicios.currval;
		SELECT * INTO servicio FROM servicios WHERE OID_S = W_OID_S;
		IF(SERVICIO.TIPOSERVICIO<>W_TipoServicio and
SERVICIO.DISTANCIAKM<>W_DistanciaKM and SERVICIO.HORASGRUA<>W_HorasGrua and
SERVICIO.TAMAÑOCUBA<>W_TamañoCuba and SERVICIO.PESOKG<>W_PesoKG AND
SERVICIO.VOLUMENM3<>W_VolumenM3) THEN
			salida:=false;
		END IF;
		COMMIT WORK;

		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));
```

```
                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END insertar;

        --PRUEBA PARA LA ACTUALIZACION DE SERVICIOS--
        PROCEDURE actualizar (nombre_prueba VARCHAR2, W_OID_S INTEGER,
W_TipoServicio VARCHAR2, W_DistanciaKM INTEGER, W_HorasGrua INTEGER,
W_TamañoCuba VARCHAR2, W_PesoKG INTEGER, W_VolumenM3 INTEGER, salidaEsperada
BOOLEAN) AS
                salida BOOLEAN := true;
                servicio servicios%ROWTYPE;
        BEGIN
                --Actualizar empleado--
                UPDATE servicios SET TipoServicio=W_TipoServicio,
DistanciaKM=W_DistanciaKM, HorasGrua=W_HorasGrua, TamañoCuba=W_TamañoCuba,
PesoKG=W_PesoKG, VolumenM3=W_VolumenM3 WHERE OID_S=W_OID_S;

                --Seleccionar gasto y comprobar que los campos se actualizaron correctamente-
-
                SELECT * INTO servicio FROM SERVICIOS WHERE OID_S = W_OID_S;
                IF(SERVICIO.TIPOSERVICIO<>W_TipoServicio and
SERVICIO.DISTANCIAKM<>W_DistanciaKM and SERVICIO.HORASGRUA<>W_HorasGrua and
SERVICIO.TAMAÑOCUBA<>W_TamañoCuba and SERVICIO.PESOKG<>W_PesoKG AND
SERVICIO.VOLUMENM3<>W_VolumenM3) THEN
                        salida:= false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END actualizar;

        --PRUEBA PARA LA ELIMINACION DE SERVICIOS--
        PROCEDURE eliminar (nombre_prueba VARCHAR2, W_OID_S INTEGER, salidaEsperada
BOOLEAN) AS
        salida BOOLEAN := true;
        n_servicios INTEGER;
        BEGIN
                --Eliminar servicio--
                DELETE FROM servicios WHERE OID_S=W_OID_S;

                --Verificar que el servicio no se encuentra en la BD--
```

```
                SELECT COUNT (*) INTO n_servicios FROM servicios WHERE
OID_S=W_OID_S;
                IF(n_servicios<>0) THEN
                        salida:=false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END eliminar;
END PRUEBAS_SERVICIOS;
/


----------------PRUEBA_FACTURAS-------------------------------------------------------------------
create or replace
PACKAGE PRUEBAS_FACTURAS AS
        PROCEDURE inicializar;
        PROCEDURE insertar
        (nombre_prueba VARCHAR2, W_NumPedidos INTEGER, W_PrecioSinIva NUMBER,
W_EstaPagado CHAR, W_FechaCreacion DATE, salidaEsperada BOOLEAN);
        PROCEDURE actualizar
        (nombre_prueba VARCHAR2, W_OID_F INTEGER, W_NumPedidos INTEGER,
W_PrecioSinIva NUMBER, W_EstaPagado CHAR, W_FechaCreacion DATE, salidaEsperada
BOOLEAN);
        PROCEDURE eliminar
        (nombre_prueba VARCHAR2, W_OID_F INTEGER, salidaEsperada BOOLEAN);

END PRUEBAS_FACTURAS;
/
--------------------CUERPO DEL PAQUETE PRUEBAS_FACTURAS--------------------
create or replace
PACKAGE BODY PRUEBAS_FACTURAS AS
        --INICIALIZACION
        PROCEDURE inicializar AS
        BEGIN
        --Borrar contenido de la tabla--
                DELETE FROM FACTURAS;
        END inicializar;

        --PRUEBA PARA LA INSERCION DE FACTURAS--
        PROCEDURE insertar    (nombre_prueba VARCHAR2, W_NumPedidos INTEGER,
W_PrecioSinIva NUMBER, W_EstaPagado CHAR, W_FechaCreacion DATE, salidaEsperada
BOOLEAN) AS
                salida BOOLEAN := true;
                factura facturas%ROWTYPE;
```

```
                W_OID_F INTEGER;
        BEGIN
                --Insertar factura--
                INSERT INTO facturas(NUMPEDIDOS,PrecioSinIva,
ESTAPAGADO,FECHACREACION) VALUES(W_NumPedidos,W_PrecioSinIva,
W_EstaPagado,W_FechaCreacion);
                --Seleccionar factura y comprobar que los datos se insertan correctamente--
                W_OID_F:= sec_facturas.currval;
                SELECT * INTO factura FROM facturas WHERE OID_F = W_OID_F;
                IF(FACTURA.NUMPEDIDOS<>W_NumPedidos and
FACTURA.PrecioSinIva<>w_PrecioSinIva and FACTURA.ESTAPAGADO<>W_EstaPagado and
FACTURA.FechaCreacion<>W_FechaCreacion) THEN
                        salida:=false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END insertar;

        --PRUEBA PARA LA ACTUALIZACION DE FACTURAS--
        PROCEDURE actualizar (nombre_prueba VARCHAR2, w_OID_F
INTEGER,W_NumPedidos INTEGER, W_PrecioSinIva NUMBER, W_EstaPagado CHAR,
W_FechaCreacion DATE, salidaEsperada BOOLEAN) AS
                salida BOOLEAN := true;
                factura facturas%ROWTYPE;
        BEGIN
                --Actualizar factura--
                UPDATE facturas SET
NumPedidos=W_NumPedidos,PrecioSinIva=W_PrecioSinIva, EstaPagado=W_EstaPagado,
FechaCreacion=W_FechaCreacion WHERE OID_F=W_OID_F;

                --Seleccionar factura y comprobar que los campos se actualizaron
correctamente--
                SELECT * INTO factura FROM FACTURAS WHERE OID_F = W_OID_F;
                IF(FACTURA.NUMPEDIDOS<>W_NumPedidos and
FACTURA.PrecioSinIva<>W_PrecioSinIva and FACTURA.ESTAPAGADO<>W_EstaPagado and
FACTURA.FechaCreacion<>W_FechaCreacion) THEN
                        salida:= false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));
```

```
                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END actualizar;

        --PRUEBA PARA LA ELIMINACION DE FACTURAS--
        PROCEDURE eliminar (nombre_prueba VARCHAR2, W_OID_F INTEGER, salidaEsperada
BOOLEAN) AS
        salida BOOLEAN := true;
        n_facturas INTEGER;
        BEGIN
                --Eliminar servicio--
                DELETE FROM facturas WHERE OID_F=W_OID_F;

                --Verificar que la factura no se encuentra en la BD--
                SELECT COUNT (*) INTO n_facturas FROM facturas WHERE OID_F=W_OID_F;
                IF(n_facturas<>0) THEN
                        salida:=false;
                END IF;
                COMMIT WORK;

                --Mostrar resultado de la prueba--
                DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

                EXCEPTION
                WHEN OTHERS THEN
                        DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
                        ROLLBACK;
        END eliminar;
END PRUEBAS_FACTURAS;
/

----------------PRUEBA_PEDIDOS--------------------------------------------------------------------------
create or replace
PACKAGE PRUEBAS_PEDIDOS AS
        PROCEDURE inicializar;
        PROCEDURE insertar
        (nombre_prueba VARCHAR2, W_precio NUMBER, W_Direccion VARCHAR2, W_Fecha
DATE, W_OID_C INTEGER, W_OID_E INTEGER, W_OID_S INTEGER, salidaEsperada
BOOLEAN);
        PROCEDURE actualizar
        (nombre_prueba VARCHAR2, W_OID_P INTEGER, W_precio NUMBER, W_Direccion
VARCHAR2, W_Fecha DATE, W_OID_C INTEGER, W_OID_E INTEGER, W_OID_S INTEGER,
salidaEsperada BOOLEAN);
        PROCEDURE eliminar
        (nombre_prueba VARCHAR2, W_OID_P INTEGER, salidaEsperada BOOLEAN);

END PRUEBAS_PEDIDOS;
/
```

```
----------CUERPO DEL PAQUETE PRUEBAS_PEDIDOS----------
create or replace
PACKAGE BODY PRUEBAS_PEDIDOS AS
	--INICIALIZACION
	PROCEDURE inicializar AS
	BEGIN
	--Borrar contenido de la tabla--
		DELETE FROM PEDIDOS;
	END inicializar;

	--PRUEBA PARA LA INSERCION DE PEDIDOS--
	PROCEDURE insertar (nombre_prueba VARCHAR2, W_precio NUMBER, W_Direccion
VARCHAR2, W_Fecha DATE, W_OID_C INTEGER, W_OID_E INTEGER, W_OID_S INTEGER,
salidaEsperada BOOLEAN) AS
		salida BOOLEAN := true;
		pedido pedidos%ROWTYPE;
		W_OID_P INTEGER;
	BEGIN
		--Insertar pedido--
		INSERT INTO pedidos(PRECIO,DIRECCION,FECHA, OID_C, OID_E, OID_S)
VALUES(W_precio, W_Direccion, W_Fecha, W_OID_C, w_OID_E, W_OID_S);
		--Seleccionar factura y comprobar que los datos se insertan correctamente--
		W_OID_P:= sec_pedidos.currval;
		SELECT * INTO pedido FROM pedidos WHERE OID_P = W_OID_P;
		IF(PEDIDO.DIRECCION<>W_Direccion and PEDIDO.FECHA<>W_Fecha and
PEDIDO.OID_C<>W_OID_C AND PEDIDO.OID_E<>w_OID_E AND
PEDIDO.OID_S<>w_OID_S) THEN
			salida:=false;
		END IF;
		COMMIT WORK;

		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

		EXCEPTION
		WHEN OTHERS THEN
			DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
			ROLLBACK;
	END insertar;

	--PRUEBA PARA LA ACTUALIZACION DE PEDIDOS--
	PROCEDURE actualizar (nombre_prueba VARCHAR2, W_OID_P INTEGER, W_precio
NUMBER, W_Direccion VARCHAR2, W_Fecha DATE, W_OID_C INTEGER, W_OID_E INTEGER,
W_OID_S INTEGER, salidaEsperada BOOLEAN) AS
		salida BOOLEAN := true;
		pedido pedidos%ROWTYPE;
	BEGIN
		--Actualizar pedido--
		UPDATE pedidos SET OID_P=W_OID_P, Precio=W_Precio,
Direccion=W_Direccion, Fecha=W_Fecha, OID_C=W_OID_C, OID_E=W_OID_E,
OID_S=W_OID_S WHERE OID_P=W_OID_P;
```

```
		--Seleccionar pedido y comprobar que los campos se actualizaron
correctamente--
		SELECT * INTO pedido FROM PEDIDOS WHERE OID_P = W_OID_P;
		IF(PEDIDO.DIRECCION<>W_Direccion and PEDIDO.FECHA<>W_Fecha and
PEDIDO.OID_C<>W_OID_C AND PEDIDO.OID_E<>W_OID_E AND
PEDIDO.OID_S<>w_OID_S) THEN
			salida:= false;
		END IF;
		COMMIT WORK;

		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

		EXCEPTION
		WHEN OTHERS THEN
			DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
			ROLLBACK;
	END actualizar;

	--PRUEBA PARA LA ELIMINACION DE PEDIDOS--
	PROCEDURE eliminar (nombre_prueba VARCHAR2, W_OID_P INTEGER, salidaEsperada
BOOLEAN) AS
	salida BOOLEAN := true;
	n_pedidos INTEGER;
	BEGIN
		--Eliminar pedido--
		DELETE FROM pedidos WHERE OID_P=W_OID_P;

		--Verificar que el pedido no se encuentra en la BD--
		SELECT COUNT (*) INTO n_pedidos FROM pedidos WHERE OID_P=W_OID_P;
		IF(n_pedidos<>0) THEN
			salida:=false;
		END IF;
		COMMIT WORK;

		--Mostrar resultado de la prueba--
		DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,
salidaEsperada));

		EXCEPTION
		WHEN OTHERS THEN
			DBMS_OUTPUT.put_line(nombre_prueba || ':' ||
ASSERT_EQUALS(false,salidaEsperada));
			ROLLBACK;
	END eliminar;
END PRUEBAS_PEDIDOS;
/
```

```
--ESTA ES LA FUNCION AUXILIAR DE LAS DIAPOSITIVAS
create or replace
FUNCTION ASSERT_EQUALS(salida BOOLEAN, salida_esperada BOOLEAN) RETURN VARCHAR2
AS
BEGIN
        IF(salida = salida_esperada) THEN
                RETURN 'EXITO';
        ELSE
                RETURN 'FALLO';
        END IF;
END ASSERT_EQUALS;
/


----------EJECUCIÓN PRUEBAS----------

SET SERVEROUTPUT ON;

DECLARE
COD_GASTOS INTEGER;

COD_PEDIDOS INTEGER;
COD_CLIENTES_PRUEBA INTEGER;
COD_EMPLEADOS_PRUEBA INTEGER;
COD_SERVICIOS_PRUEBA INTEGER;

COD_CLIENTES INTEGER;

COD_FACTURAS INTEGER;

COD_EMPLEADOS INTEGER;

COD_SERVICIOS INTEGER;



BEGIN

----------PRUEBAS GASTOS

PRUEBAS_GASTOS.INICIALIZAR;

PRUEBAS_GASTOS.INSERTAR('PRUEBA 1 - INSERCIÓN GASTOS', 'Gasoil', 'Pago de gasoil', 500,
'21/1/2015', '1/1/15', '31/1/15', true);
COD_GASTOS:= SEC_GASTOS.CURRVAL;

PRUEBAS_GASTOS.INSERTAR('PRUEBA 2 - INSERCIÓN GASTOS nombre en null', null, 'Pago de
gasoil', 500, '21/1/2015', '1/1/15', '31/1/15', false);
PRUEBAS_GASTOS.INSERTAR('PRUEBA 3 - INSERCIÓN GASTOS descripcion null', 'Gasoil', null,
500, '21/1/2015', '1/1/15', '31/1/15', false);
```

```
PRUEBAS_GASTOS.INSERTAR('PRUEBA 4 - INSERCIÓN GASTOS coste null', 'Gasoil', 'Pago de gasoil', null, '21/1/2015', '1/1/15', '31/1/15', false);
PRUEBAS_GASTOS.INSERTAR('PRUEBA 5 - INSERCIÓN GASTOS fecha pago null', 'Gasoil', 'Pago de gasoil', 500, null, '1/1/15', '31/1/15', false);
PRUEBAS_GASTOS.INSERTAR('PRUEBA 6 - INSERCIÓN GASTOS fecha inicio null', 'Gasoil', 'Pago de gasoil', 500, '21/1/2015', null, '31/1/15', false);
PRUEBAS_GASTOS.INSERTAR('PRUEBA 7 - INSERCIÓN GASTOS fecha fin null', 'Gasoil', 'Pago de gasoil', 500, '21/1/2015', '1/1/15', null, false);
PRUEBAS_GASTOS.INSERTAR('PRUEBA 8 - INSERCIÓN GASTOS fecha fin antes que fecha inicio', 'Gasoil', 'Pago de gasoil', 500, '21/1/2015', '1/2/15', '31/1/15', false);


PRUEBAS_GASTOS.actualizar('PRUEBA 9 - ACTUALIZACIÓN GASTOS',COD_GASTOS, 'Gasolina', 'Pago de gasolina', 600, '21/1/2016', '1/1/16', '31/1/16', true);


PRUEBAS_GASTOS.actualizar('PRUEBA 10 - ACTUALIZACIÓN GASTOS nombre en null', COD_GASTOS, null, 'Pago de gasoil', 500, '21/1/2015', '1/1/15', '31/1/15', false);
PRUEBAS_GASTOS.actualizar('PRUEBA 11 - ACTUALIZACIÓN GASTOS descripcion null',COD_GASTOS, 'Gasoil', null, 500, '21/1/2015', '1/1/15', '31/1/15', false);
PRUEBAS_GASTOS.actualizar('PRUEBA 12 - ACTUALIZACIÓN GASTOS coste null',COD_GASTOS, 'Gasoil', 'Pago de gasoil', null, '21/1/2015', '1/1/15', '31/1/15', false);
PRUEBAS_GASTOS.actualizar('PRUEBA 13 - ACTUALIZACIÓN GASTOS fecha pago null',COD_GASTOS, 'Gasoil', 'Pago de gasoil', 500, null, '1/1/15', '31/1/15', false);
PRUEBAS_GASTOS.actualizar('PRUEBA 14 - ACTUALIZACIÓN GASTOS fecha inicio null',COD_GASTOS, 'Gasoil', 'Pago de gasoil', 500, '21/1/2015', null, '31/1/15', false);
PRUEBAS_GASTOS.actualizar('PRUEBA 15 - ACTUALIZACIÓN GASTOS fin null',COD_GASTOS, 'Gasoil', 'Pago de gasoil', 500, '21/1/2015', '1/1/15', null, false);
PRUEBAS_GASTOS.actualizar('PRUEBA 16 - ACTUALIZACIÓN GASTOS fecha fin antes que fecha inicio',COD_GASTOS, 'Gasoil', 'Pago de gasoil', 500, '21/1/2015', '1/2/15', '31/1/15', false);
PRUEBAS_GASTOS.actualizar('PRUEBA 17 - ACTUALIZACIÓN GASTOS oid_g a null',null, 'Gasoil', 'Pago de gasoil', 500, '21/1/2015', '1/1/15', '31/1/15', false);

PRUEBAS_GASTOS.ELIMINAR('PRUEBA 18 - ELIMINACIÓN GASTOS', COD_GASTOS, true);


----------------------------------------------------------------PRUEBAS CLIENTES

PRUEBAS_CLIENTES.INICIALIZAR;
PRUEBAS_CLIENTES.INSERTAR('PRUEBA 1 - INSERCIÓN CLIENTES buena', 'Juan', 'C/Clavel s/n', '123456789', 'juan@prueba.com', true);
COD_CLIENTES:= sec_clientes.currval;
PRUEBAS_CLIENTES.INSERTAR('PRUEBA 1.2 - INSERCIÓN CLIENTES buena', 'Juan', 'C/Clavel s/n', '123456789', 'juan2@prueba.com', true);


PRUEBAS_CLIENTES.INSERTAR('PRUEBA 2 - INSERCIÓN CLIENTES nombre a null', null, 'C/Clavel s/n', '123456789', 'juan@prueba.com', false);
PRUEBAS_CLIENTES.INSERTAR('PRUEBA 3 - INSERCIÓN CLIENTES telefono a null', 'Juan', 'C/Clavel s/n', null, 'juan@prueba.com', false);
PRUEBAS_CLIENTES.INSERTAR('PRUEBA 4 - INSERCIÓN CLIENTES correo a null', 'Juan', 'C/Clavel s/n', '123456789', null, false);
```

```
PRUEBAS_CLIENTES.INSERTAR('PRUEBA 5 - INSERCIÓN CLIENTES mismo correo', 'Juan',
'C/Clavel s/n', '123456789', 'juan@prueba.com', false);

PRUEBAS_CLIENTES.ACTUALIZAR('PRUEBA 6  - ACTUALIZACIÓN CLIENTES', COD_CLIENTES,
'Juan Andres', 'C/Clavel 6', '223456789', 'juanito@prueba.com', true);
PRUEBAS_CLIENTES.ACTUALIZAR('PRUEBA 7  - ACTUALIZACIÓN CLIENTES oid_c a null', null,
'Juan Andres', 'C/Clavel 6', '223456789', 'juan2@prueba.com', false);
PRUEBAS_CLIENTES.ACTUALIZAR('PRUEBA 8  - ACTUALIZACIÓN CLIENTES nombre a null',
COD_CLIENTES, null, 'C/Clavel s/n', '123456789', 'juan@prueba.com', false);
PRUEBAS_CLIENTES.ACTUALIZAR('PRUEBA 9  - ACTUALIZACIÓN CLIENTES telefono a null',
COD_CLIENTES, 'Juan Andres', 'C/Clavel s/n', null, 'juan@prueba.com', false);
PRUEBAS_CLIENTES.ACTUALIZAR('PRUEBA 10  - ACTUALIZACIÓN CLIENTES correo a null',
COD_CLIENTES, 'Juan Andres', 'C/Clavel s/n', '123456789', null, false);
PRUEBAS_CLIENTES.ACTUALIZAR('PRUEBA 11  - ACTUALIZACIÓN CLIENTES mismo correo',
COD_CLIENTES, 'Juan Andres', 'C/Clavel s/n', '123456789', 'juan2@prueba.com', false);

PRUEBAS_CLIENTES.ELIMINAR('PRUEBA 12 - ELIMINACIÓN CLIENTES', COD_CLIENTES, true);


------------------------------------------------------------PRUEBAS FACTURAS


PRUEBAS_FACTURAS.INICIALIZAR;
PRUEBAS_FACTURAS.INSERTAR('PRUEBA 1 - INSERCIÓN FACTURA buena', 4, 140, 'NO',
'21/1/2018', true);
COD_FACTURAS:= sec_facturas.currval;

PRUEBAS_FACTURAS.INSERTAR('PRUEBA 2 - INSERCIÓN FACTURA fecha emision antes fecha
creacion', 4, 140, 'SI', '21/1/2018', false);
PRUEBAS_FACTURAS.INSERTAR('PRUEBA 3 - INSERCIÓN FACTURA numero pedidos a null',
null, 140, 'SI', '21/1/2012', false);
PRUEBAS_FACTURAS.INSERTAR('PRUEBA 4 - INSERCIÓN FACTURA esta pagado a null', 4, 140,
null, '21/1/2018', false);
PRUEBAS_FACTURAS.INSERTAR('PRUEBA 5 - INSERCIÓN FACTURA esta pagado invalido', 4,
140, 'moroso', '21/1/2018', false);
PRUEBAS_FACTURAS.INSERTAR('PRUEBA 6 - INSERCIÓN FACTURA precio a null', 4, null, 'SI',
'21/1/2018', false);


PRUEBAS_FACTURAS.ACTUALIZAR('PRUEBA 6 - ACTUALIZACION FACTURA buena',
COD_FACTURAS, 3, 140, 'NO', '10/6/2018', true);
PRUEBAS_FACTURAS.ACTUALIZAR('PRUEBA 7 - ACTUALIZACION FACTURA oid_f a null', null, 3,
140, 'SI', '10/6/2018', false);
PRUEBAS_FACTURAS.ACTUALIZAR('PRUEBA 8 - ACTUALIZACION FACTURA fecha emision antes
fecha creacion', COD_FACTURAS, 3, 140, 'SI', '10/6/2018', false);
PRUEBAS_FACTURAS.ACTUALIZAR('PRUEBA 9 - ACTUALIZACION FACTURA esta pagado a null',
COD_FACTURAS, 3, 140, null, '10/6/2018', false);
PRUEBAS_FACTURAS.ACTUALIZAR('PRUEBA 10 - ACTUALIZACION FACTURA esta pagado
invalido', COD_FACTURAS, 3, 140, 'moroso', '10/6/2018', false);
PRUEBAS_FACTURAS.ACTUALIZAR('PRUEBA 11 - ACTUALIZACION FACTURA precio a
null',COD_FACTURAS, 4, null, 'SI', '21/1/2018', false);
```

```
PRUEBAS_FACTURAS.ELIMINAR('PRUEBA 12 - ELIMINACIÓN CLIENTES', COD_FACTURAS,
true);


----------PRUEBAS EMPLEADOS


PRUEBAS_EMPLEADOS.INICIALIZAR;

PRUEBAS_EMPLEADOS.INSERTAR('PRUEBA 1 - INSERCIÓN EMPLEADOS', 'Paco' , 'Camion',
true);
COD_EMPLEADOS:=SEC_EMPLEADOS.CURRVAL;

PRUEBAS_EMPLEADOS.INSERTAR('PRUEBA 2 - INSERCIÓN EMPLEADOS - Nombre empleado a
null', null , 'Camion', false);
PRUEBAS_EMPLEADOS.INSERTAR('PRUEBA 3 - INSERCIÓN EMPLEADOS - Tipo camion a null ',
'Paco' , null, false);
PRUEBAS_EMPLEADOS.INSERTAR('PRUEBA 4 - INSERCIÓN EMPLEADOS - Tipo camion
inválido', 'Paco' , 'Tractor', false);

PRUEBAS_EMPLEADOS.ACTUALIZAR('PRUEBA 5 - ACTUALIZACIÓN EMPLEADOS',
COD_EMPLEADOS, 'Manuel', 'Camion_Grua', true);
PRUEBAS_EMPLEADOS.ACTUALIZAR('PRUEBA 6 - ACTUALIZACIÓN EMPLEADOS - Nombre
empleado a null', COD_EMPLEADOS, null, 'Camion_Grua', false);
PRUEBAS_EMPLEADOS.ACTUALIZAR('PRUEBA 7 - ACTUALIZACIÓN EMPLEADOS - Tipo camion
a null', COD_EMPLEADOS, 'Manuel', null, false);
PRUEBAS_EMPLEADOS.ACTUALIZAR('PRUEBA 8 - ACTUALIZACIÓN EMPLEADOS - Tipo camion
inválido', COD_EMPLEADOS, 'Manuel', 'Tractor', false);

PRUEBAS_EMPLEADOS.ELIMINAR('PRUEBA 9 - ELIMINACIÓN
EMPLEADOS',COD_EMPLEADOS,TRUE);


----------PRUEBAS SERVICIOS

PRUEBAS_SERVICIOS.INICIALIZAR;

PRUEBAS_SERVICIOS.INSERTAR('PRUEBA 1 - INSERCIÓN SERVICIOS', 'CUBA_ESCOMBROS',
30, NULL, 'PEQUEÑA', NULL,NULL, TRUE);
COD_SERVICIOS:=SEC_SERVICIOS.CURRVAL;

PRUEBAS_SERVICIOS.INSERTAR('PRUEBA 2 - INSERCIÓN SERVICIOS - Tipo servicio null',
NULL, 30, NULL, 'PEQUEÑA', NULL,NULL, FALSE);
PRUEBAS_SERVICIOS.INSERTAR('PRUEBA 2 - INSERCIÓN SERVICIOS - Tipo servicio inválido',
'PRIEDRA_CALIZA', 30, NULL, 'PEQUEÑA', NULL,NULL, FALSE);
PRUEBAS_SERVICIOS.INSERTAR('PRUEBA 3 - INSERCIÓN SERVICIOS - Distancia mayor de 35
km', 'CUBA_ESCOMBROS', 40, NULL, 'PEQUEÑA', NULL,NULL, FALSE);
PRUEBAS_SERVICIOS.INSERTAR('PRUEBA 4 - INSERCIÓN SERVICIOS - Tamaño cuba inválido',
'CUBA_ESCOMBROS', 30, NULL, 'MUY GRANDE', NULL,NULL, FALSE);
PRUEBAS_SERVICIOS.INSERTAR('PRUEBA 5 - INSERCIÓN SERVICIOS - Peso mayor de 4500',
'LEÑA', NULL, NULL, NULL, 5000,NULL, FALSE);
PRUEBAS_SERVICIOS.INSERTAR('PRUEBA 6 - INSERCIÓN SERVICIOS - Volumen mayor de 5
m3', 'ÁRIDOS', 30, NULL, NULL, NULL, 7, FALSE);
```

```
PRUEBAS_SERVICIOS.ACTUALIZAR('PRUEBA 7 - ACTUALIZACIÓN SERVICIOS',
COD_SERVICIOS, 'CUBA_PODA', 30, NULL, 'PEQUEÑA', NULL,NULL, TRUE);
PRUEBAS_SERVICIOS.ACTUALIZAR('PRUEBA 8 - ACTUALIZACIÓN SERVICIOS - Tipo servicio
null', COD_SERVICIOS, NULL, 30, NULL, 'PEQUEÑA', NULL,NULL, FALSE);
PRUEBAS_SERVICIOS.ACTUALIZAR('PRUEBA 9 - ACTUALIZACIÓN SERVICIOS - Tipo servicio
inválido', COD_SERVICIOS, 'PRIEDRA_CALIZA', 30, NULL, 'PEQUEÑA', NULL,NULL, FALSE);
PRUEBAS_SERVICIOS.ACTUALIZAR('PRUEBA 10 - ACTUALIZACIÓN SERVICIOS - Distancia
mayor de 35 km', COD_SERVICIOS, 'CUBA_ESCOMBROS', 40, NULL, 'PEQUEÑA', NULL,NULL,
FALSE);
PRUEBAS_SERVICIOS.ACTUALIZAR('PRUEBA 11 - ACTUALIZACIÓN SERVICIOS - Tamaño cuba
inválido', COD_SERVICIOS, 'CUBA_ESCOMBROS', 30, NULL, 'MUY GRANDE', NULL,NULL,
FALSE);
PRUEBAS_SERVICIOS.ACTUALIZAR('PRUEBA 12 - ACTUALIZACIÓN SERVICIOS - Peso mayor de
4500', COD_SERVICIOS, 'LEÑA', NULL, NULL, NULL, 5000,NULL, FALSE);
PRUEBAS_SERVICIOS.ACTUALIZAR('PRUEBA 13 - ACTUALIZACIÓN SERVICIOS - Volumen
mayor de 5 m3', COD_SERVICIOS, 'ÁRIDOS', 30, NULL, NULL, NULL, 7, FALSE);

PRUEBAS_SERVICIOS.ELIMINAR('PRUEBA 14 - ELIMINACIÓN SERVICIOS', COD_SERVICIOS,
TRUE);



------------------------------------------------------------PRUEBAS PEDIDOS

PRUEBAS_CLIENTES.INICIALIZAR;
INSERT INTO CLIENTES(NOMBRE, DIRECCION, NUMEROTELEFONO, CORREOELECTRONICO)
VALUES('Juan', 'C/Clavel s/n', '123456789', 'orion@prueba.com');
COD_clientes_PRUEBA :=SEC_CLIENTES.CURRVAL;

PRUEBAS_EMPLEADOS.INICIALIZAR;
INSERT INTO EMPLEADOS(NOMBREE, TIPOCAMION) VALUES('PACO', 'Camion');
COD_EMPLEADOS_PRUEBA:=SEC_EMPLEADOS.CURRVAL;


PRUEBAS_SERVICIOS.INICIALIZAR;
INSERT INTO SERVICIOS(TIPOSERVICIO, DISTANCIAKM, HORASGRUA, TAMAÑOCUBA,
PESOKG, VOLUMENM3) VALUES ('CUBA_ESCOMBROS', 30, NULL, 'PEQUEÑA', NULL,NULL);
COD_SERVICIOS_PRUEBA:=SEC_SERVICIOS.CURRVAL;

PRUEBAS_PEDIDOS.INICIALIZAR;

PRUEBAS_PEDIDOS.INSERTAR('PRUEBA 1 - INSERCIÓN PEDIDOS', 500, 'CALLE LUIS DAOIZ',
'20/09/14', COD_CLIENTES_PRUEBA, COD_EMPLEADOS_PRUEBA, COD_SERVICIOS_PRUEBA,
true);
COD_PEDIDOS:= SEC_PEDIDOS.CURRVAL;

PRUEBAS_PEDIDOS.INSERTAR('PRUEBA 2 - INSERCIÓN PEDIDOS precio null', NULL, 'CALLE
LUIS DAOIZ', '20/09/14', COD_CLIENTES_PRUEBA, COD_EMPLEADOS_PRUEBA,
COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.INSERTAR('PRUEBA 3 - INSERCIÓN PEDIDOS dirección null', 500, NULL,
'20/09/14', COD_CLIENTES_PRUEBA, COD_EMPLEADOS_PRUEBA, COD_SERVICIOS_PRUEBA,
false);
```

```
PRUEBAS_PEDIDOS.INSERTAR('PRUEBA 4 - INSERCIÓN PEDIDOS fecha null', 500, 'CALLE LUIS
DAOIZ', NULL, COD_CLIENTES_PRUEBA, COD_EMPLEADOS_PRUEBA,
COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.INSERTAR('PRUEBA 5 - INSERCIÓN PEDIDOS oid_c null', 500, 'CALLE LUIS
DAOIZ', '20/09/14', NULL, COD_EMPLEADOS_PRUEBA, COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.INSERTAR('PRUEBA 6 - INSERCIÓN PEDIDOS oid_e null', 500, 'CALLE LUIS
DAOIZ', '20/09/14', COD_CLIENTES_PRUEBA, NULL, COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.INSERTAR('PRUEBA 7 - INSERCIÓN PEDIDOS oid_s null', 500, 'CALLE LUIS
DAOIZ', '20/09/14', COD_CLIENTES_PRUEBA, COD_EMPLEADOS_PRUEBA, NULL, false);

PRUEBAS_PEDIDOS.ACTUALIZAR('PRUEBA 8 - ACTUALIZACIÓN PEDIDOS', COD_PEDIDOS,
600, 'CALLE LUIS MONTOTO', '20/09/15', COD_CLIENTES_PRUEBA,
COD_EMPLEADOS_PRUEBA, COD_SERVICIOS_PRUEBA, true);

PRUEBAS_PEDIDOS.ACTUALIZAR('PRUEBA 9 - ACTUALIZACIÓN PEDIDOS precio null',
COD_PEDIDOS,NULL, 'CALLE LUIS DAOIZ', '20/09/14', COD_CLIENTES_PRUEBA,
COD_EMPLEADOS_PRUEBA, COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.ACTUALIZAR('PRUEBA 10 - ACTUALIZACIÓN PEDIDOS dirección null',
COD_PEDIDOS,500, NULL, '20/09/14', COD_CLIENTES_PRUEBA, COD_EMPLEADOS_PRUEBA,
COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.ACTUALIZAR('PRUEBA 11 - ACTUALIZACIÓN PEDIDOS fecha null',
COD_PEDIDOS,500, 'CALLE LUIS DAOIZ', NULL, COD_CLIENTES_PRUEBA,
COD_EMPLEADOS_PRUEBA, COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.ACTUALIZAR('PRUEBA 12 - ACTUALIZACIÓN PEDIDOS oid_c null',
COD_PEDIDOS,500, 'CALLE LUIS DAOIZ', '20/09/14', NULL, COD_EMPLEADOS_PRUEBA,
COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.ACTUALIZAR('PRUEBA 13 - ACTUALIZACIÓN PEDIDOS oid_e null',
COD_PEDIDOS,500, 'CALLE LUIS DAOIZ', '20/09/14', COD_CLIENTES_PRUEBA, NULL,
COD_SERVICIOS_PRUEBA, false);
PRUEBAS_PEDIDOS.ACTUALIZAR('PRUEBA 14 - ACTUALIZACIÓN PEDIDOS oid_s null',
COD_PEDIDOS,500, 'CALLE LUIS DAOIZ', '20/09/14', COD_CLIENTES_PRUEBA,
COD_EMPLEADOS_PRUEBA, NULL, false);

PRUEBAS_PEDIDOS.ELIMINAR('PRUEBA 15 - ELIMINACIÓN PEDIDOS', COD_PEDIDOS, true);



END;
```