**Diseño**

**Documento de Diseño**

**Área (REQM)**

Sistema de Delivery Online

Diseño

Actualizado a Julio del 2014

PROSOFT

1. **Introducción**

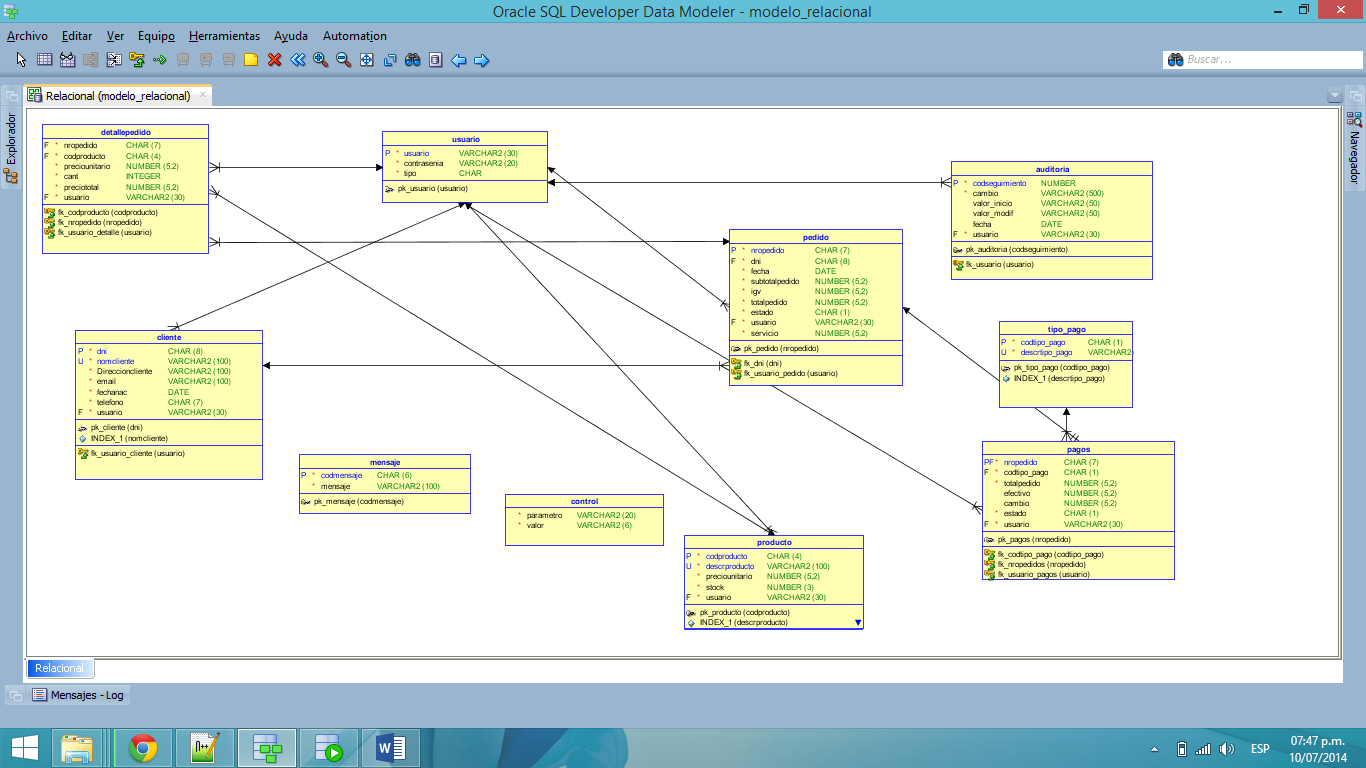
El presente documento describe el diseño externo del Sistema de Delivery Online. Este diseño incluye el diseño de interfaces gráficas (prototipos), diseño de procedimientos (rutinas que no tienen interfaz gráfica), diseño de reportes que puedan complementar el funcionamiento y objetivo de cada diseño. Cada diseño de presente documento tiene una descripción detallada de lo que pretende implementar, esta descripción se divide en campos, botones de acción, perfiles, flujo de eventos, reglas de negocio y validaciones de campos.

El diseño tiene como objetivo general ser una guía detallada de cómo se van a implementar los requisitos del sistema y de cómo el usuario final lo va a percibir sin entrar a detalles técnicos de implementación. En otras palabras, describe la funcionalidad esperada por el usuario final.

1. **Descripción general del sistema**

El sistema completo considera dos usuarios: el usuario final y el administrador. Y para cada uno de estos crear una aplicación acorde a las tareas que realizará cada uno, y los permisos que tenga. Para el usuario final es necesaria una aplicación cliente que muestre el registro de datos del usuario en la web de la pizzería y obtener una cuenta, el módulo de pedido mostrara las categorías y platos que ofrece la pizzería lo cual permitirá al usuario agregar dichos platos a un carro de compras y efectuar el pedido, con los ítems seleccionados. También cuenta con el módulo de pago que se hará de forma presencial ya sea con tarjeta o dinero en efectivo.

La aplicación del administrador del restaurante requiere mayor trabajo ya que no sólo abarca el agregar, modificar y borrar platos del restaurante. Además se encuentran involucrados la administración de repartidores, modificación de pedidos, estadísticas y modelo de negocio del reparto.

1. **Tablas de Base de datos**

-- ====================================

-- Crear la Tabla de Usuarios:usuario

-- ====================================

create table usuario(

usuario varchar2(30) not null,

contrasenia varchar2(20) not null,

tipo char not null,

constraint pk\_usuario

primary key(usuario)

);

-- =====================================

-- Crear la Tabla de Productos:producto

-- =====================================

create table producto(

codproducto char(4) not null,

nombreproducto varchar2(100) not null,

descrproducto varchar2(400)not null,

categoria varchar2(30) not null,

imagen varchar2(200) not null,

preciounitario number(5,2) not null,

stock number(3,0) not null,

usuario varchar2(30) not null,

constraint pk\_producto

primary key (codproducto),

constraint fk\_usuario\_producto

foreign key (usuario)

references usuario,

constraint u\_descr\_producto

unique(descrproducto),

constraint chk\_producto\_codproducto

check( REGEXP\_LIKE(codproducto,'[PE][INS][0-9][0-9]')),

constraint chk\_producto\_preciounitario

check(preciounitario>0),

constraint chk\_producto\_stock

check(stock>=0)

);

-- ===================================

-- Crear la Tabla de Clientes:cliente

-- ===================================

create table cliente(

dni char(8) not null,

nomcliente varchar2(100) not null,

direccioncliente varchar2(100) not null,

email varchar2(100) not null,

fechanac date not null,

telefono char(7) not null,

usuario varchar2(30) not null,

constraint pk\_cliente

primary key(dni),

constraint fk\_usuario\_cliente

foreign key (usuario)

references usuario,

constraint u\_nomcliente

unique(nomcliente)

);

-- ===================================

-- Crear la Tabla de Pedidos:pedido

-- ===================================

create table pedido(

nropedido char(7) not null,

dni char(8) not null,

fecha date not null,

subtotalpedido number(5,2) not null,

igv number(5,2) not null,

totalpedido number(5,2) not null,

estado varchar2(3) not null,

usuario varchar2(30) not null,

constraint pk\_pedido

primary key(nropedido),

constraint fk\_dni

foreign key(dni)

references cliente,

constraint fk\_usuario\_pedido

foreign key (usuario)

references usuario,

constraint chk\_subtotal

check(subtotalpedido>=0.0),

constraint chk\_igv

check(igv>=0.0),

constraint chk\_totalpedido

check(totalpedido>=0.0)

);

-- ====================================================

-- Crear la Tabla de Detalle de Pedido:detallepedido

-- ====================================================

create table detallepedido(

nropedido char(7) not null,

codproducto char(4) not null,

preciounitario number(5,2) not null,

cant int not null,

preciototal number(5,2) not null,

usuario varchar2(30) not null,

constraint fk\_nropedido

foreign key (nropedido)

references pedido,

constraint fk\_codproducto

foreign key (codproducto)

references producto,

constraint fk\_usuario\_detalle

foreign key (usuario)

references usuario,

constraint chk\_detpedido\_preuni

check(preciounitario>0),

constraint chk\_detpedido\_cant

check(cant>=0),

constraint chk\_detpedido\_pretot

check(preciounitario>0)

);

-- ====================================

-- Crear la Tabla de Control:control

-- ====================================

create table control(

parametro varchar2(20) not null,

valor varchar2(6) not null

);

-- ====================================

-- Crear la tabla de Auditoria

-- ====================================

create table auditoria(

codseguimiento number not null,

cambio varchar2(500) not null,

valor\_inicio varchar2(50) ,

valor\_modif varchar2(50) ,

fecha date,

usuario varchar2(30)not null,

constraint pk\_auditoria

primary key(codseguimiento),

constraint fk\_usuario

foreign key(usuario)

references usuario

);

-- ========================================

-- Crear la tabla de Mensaje

-- ========================================

create table mensaje(

codmensaje char(6) not null,

mensaje varchar2(100) not null,

constraint pk\_mensaje

primary key(codmensaje)

);

--===================================================

--Crear tabla Tipo\_Pago

--===================================================

create table tipo\_pago(

codtipo\_pago char(1) not null,

descrtipo\_pago varchar2(50) not null,

constraint pk\_tipo\_pago

primary key(codtipo\_pago),

constraint u\_descrtipo\_pago

unique(descrtipo\_pago)

);

--===================================================

--Crear la Tabla Pagos

--===================================================

create table pagos(

nropedido char(7) not null,

codtipo\_pago char(1) not null,

totalpedido number(5,2) not null,

efectivo number(5,2),

cambio number(5,2),

estado char(1) not null,

usuario varchar2(30) not null,

constraint pk\_pagos

primary key(nropedido),

constraint fk\_nropedidos

foreign key(nropedido)

references pedido,

constraint fk\_codtipo\_pago

foreign key(codtipo\_pago)

references tipo\_pago,

constraint fk\_usuario\_pagos

foreign key (usuario)

references usuario,

constraint chk\_totalpedidos

check(totalpedido > 0)

);

1. **Procesos Almacenados**

CREATE OR REPLACE PROCEDURE sp\_agregarpago

(

p\_nropedido in varchar2,

p\_codtipo\_pago in pagos.codtipo\_pago%type,

p\_totalpedido in number,

p\_efectivo in number,

p\_cambio in number,

p\_estado pagos.estado%type,

p\_usuario in pagos.usuario%type

)

AS

v\_cont number;

v\_msg varchar2(100);

BEGIN

--Verifica pago

select count(\*) into v\_cont

from pagos

where nropedido=p\_nropedido;

if v\_cont > 0 then

raise\_application\_error(-20000, 'Pago ya existe');

end if;

insert into pagos(nropedido, codtipo\_pago, totalpedido,

efectivo, cambio, estado,usuario)

values(p\_nropedido,p\_codtipo\_pago, p\_totalpedido,

p\_efectivo, p\_cambio, p\_estado,p\_usuario);

commit;

exception

when others then

v\_msg:=sqlerrm;

rollback;

raise\_application\_error(-20000, v\_msg);

END SP\_ADD\_PAGOS;

/

create or replace procedure sp\_insertadetallepedido(

p\_nropedido PEDIDO.NROPEDIDO%TYPE,

p\_codproducto varchar2,

p\_cant number,

p\_usuario varchar2

)

as

v\_preciouni number(5,2);

v\_preciototal number(5,2);

v\_subtotal number(5,2);

v\_igv number(5,2);

v\_total number(5,2);

msg\_error varchar2(500);

v\_cant number;

begin

v\_preciouni := fn\_obtenerpreunitproducto(p\_codproducto);

v\_preciototal := v\_preciouni \* p\_cant;

if fn\_isduplicadoproducto(p\_nropedido,p\_codproducto) = 1 then

v\_cant := fn\_obtenercantdetallepedido(p\_nropedido,p\_codproducto);

v\_cant := v\_cant + p\_cant;

update detallepedido set cant = v\_cant where nropedido = p\_nropedido and codproducto = p\_codproducto;

else

insert into detallepedido values (p\_nropedido,p\_codproducto,v\_preciouni,p\_cant,v\_preciototal,p\_usuario);

end if;

v\_subtotal:= fn\_obtenersubtotalpedido(p\_nropedido);

v\_igv := v\_subtotal \* 0.18;

v\_total := v\_subtotal + v\_igv;

update pedido set subtotalpedido=v\_subtotal,igv=v\_igv,totalpedido=v\_total where nropedido = p\_nropedido;

commit;

exception

when others then

msg\_error := SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, msg\_error);

rollback;

end;

/

create or replace PROCEDURE sp\_insertacliente(

p\_dni varchar2,

p\_nombrecliente varchar2,

p\_direccioncliente varchar2,

P\_email varchar2,

p\_fechanac date,

p\_telefono varchar2,

p\_usuario varchar2,

p\_contraseña varchar2

)

AS

v\_msg\_error varchar2(500);

v\_msg varchar2(500);

n int;

val int;

err\_cli exception;

err\_usu exception;

BEGIN

/\*comprobar si existe el usuario\*/

select count(\*) into n from usuario where usuario=p\_usuario;

if n = 0 then

insert into usuario values (p\_usuario,p\_contraseña,'C');

else

raise err\_usu;

end if;

/\*comprobar si existe el cliente\*/

select count(\*) into n from cliente where dni=p\_dni;

if n = 0 then

insert into cliente values (p\_dni,p\_nombrecliente,p\_direccioncliente,P\_email,p\_fechanac,p\_telefono,p\_usuario);

else

raise err\_cli;

end if;

commit;

EXCEPTION

WHEN err\_cli THEN

v\_msg\_error:='error de insercion cliente';

raise\_application\_error(-20001,v\_msg\_error);

WHEN err\_usu THEN

v\_msg\_error:='error de insercion usuario';

raise\_application\_error(-20002,v\_msg\_error);

WHEN OTHERS THEN

v\_msg\_error:=SQLERRM;

raise\_application\_error(-20003,v\_msg\_error);

rollback;

END;

/

CREATE OR REPLACE PROCEDURE sp\_eliminarpedido(

p\_nropedido PEDIDO.NROPEDIDO%TYPE,

p\_usuario varchar2

)

as

msg\_error varchar2(500);

begin

delete from detallepedido where nropedido = p\_nropedido;

delete from pedido where nropedido = p\_nropedido;

commit;

exception

when others then

msg\_error:=SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, msg\_error);

rollback;

end;

/

CREATE OR REPLACE PROCEDURE sp\_insertapedidos(

p\_dni varchar2,

p\_usuario varchar2

)

as

v\_nro varchar2(10);

v\_inicio varchar2(7);

v\_temporal varchar2(10);

n number;

msg\_error varchar2(100);

begin

v\_nro := fn\_obtenervalorcontrol('pedido');

insert into pedido values (v\_nro,P\_dni,sysdate,0,0,0,'E',p\_usuario);

v\_inicio:=substr(v\_nro,4,3);

n:=to\_number(v\_inicio);

n:=n+1;

v\_temporal:='PED'||to\_char(n,'FM009');

update control

set valor=v\_temporal

where parametro='pedido';

commit;

exception

when others then

msg\_error:=SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, msg\_error);

rollback;

end;

/

1. **Funciones**

create or replace function fn\_obtenerpreunitproducto(

p\_codproducto varchar2

)

return number

is

v\_preunit number;

msg\_error varchar2(500);

begin

select preciounitario into v\_preunit from producto where codproducto = p\_codproducto;

return(v\_preunit);

exception

when others then

msg\_error := SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, msg\_error);

end;

/

CREATE OR REPLACE FUNCTION fn\_obtenersubtotalpedido(

p\_nropedido PEDIDO.NROPEDIDO%TYPE

)

return NUMBER

is

subtotal number(5,2);

v\_msg\_error varchar2(100);

n number;

err\_ped exception;

begin

select count(\*)into n from detallepedido where nropedido = p\_nropedido;

if n>0 then

select sum(preciototal) into subtotal from detallepedido where nropedido = p\_nropedido;

return(subtotal);

else

raise err\_ped;

end if;

exception

WHEN err\_ped THEN

v\_msg\_error:='error de busqueda de pedido';

raise\_application\_error(-20001,v\_msg\_error);

when others then

v\_msg\_error:=SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, v\_msg\_error);

end;

/

create or replace function fn\_obtenervalorcontrol(

p\_parametro varchar2

)

return varchar2

is

v\_valor varchar2(10);

msg\_error varchar2(500);

begin

select valor into v\_valor from control where parametro = p\_parametro;

return(v\_valor);

exception

when others then

msg\_error:=SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, msg\_error);

end;

/

create or replace function fn\_obtenercantdetallepedido(

p\_nropedido PEDIDO.NROPEDIDO%TYPE,

p\_codproducto varchar2

)

return number

is

v\_cant number;

msg\_error varchar2(500);

begin

select cant into v\_cant from detallepedido where codproducto = p\_codproducto and nropedido = p\_nropedido;

return(v\_cant);

exception

when others then

msg\_error := SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, msg\_error);

end;

/

create or replace function fn\_isduplicadoproducto(

p\_nropedido PEDIDO.NROPEDIDO%TYPE,

p\_codproducto varchar2

)

return number

is

v\_cant number;

msg\_error varchar2(500);

begin

select count(\*) into v\_cant from detallepedido where codproducto = p\_codproducto and nropedido = p\_nropedido;

if v\_cant > 0 then

return 1;

else

return 0;

end if;

exception

when others then

msg\_error := SQLERRM;

RAISE\_APPLICATION\_ERROR(-20003, msg\_error);

end;

/

1. **Disparadores**

createorreplace

trigger TR\_IN\_DE\_AC\_CLIENTE

AFTER INSERT OR DELETE OR UPDATE ON CLIENTE

FOR EACH ROW

DECLARE

v\_msgvarchar2(100);

BEGIN

If updating then

if :NEW.usuario IS NULL then

RAISE\_APPLICATION\_ERROR(-20000,'No se puede insertar usuario nulo');

End if;

if updating('dni') then

RAISE\_APPLICATION\_ERROR(-20001,'No se puede modificar atributo DNI');

End if;

if updating('nomcliente') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_CLIENTE campo: nomcliente',

:OLD.nomcliente,:NEW.nomcliente,:NEW.usuario,sysdate);

end if;

if updating('direccioncliente') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_CLIENTE campo: direccioncliente',

:OLD.direccioncliente,:NEW.direccioncliente,:NEW.usuario,sysdate);

end if;

if updating('email') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_CLIENTE campo: email',

:OLD.email, :NEW.email,:NEW.usuario,sysdate);

end if;

if updating('fechanac') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_CLIENTE campo: fechanac',

:OLD.fechanac, :NEW.fechanac,:NEW.usuario,sysdate);

end if;

if updating('telefono') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_CLIENTE campo: telefono',

:OLD.telefono, :NEW.telefono,:NEW.usuario,sysdate);

end if;

Elsif inserting then

If :NEW.usuario IS NULL then

RAISE\_APPLICATION\_ERROR(-20002,'No se puede eliminar registro con usuario nulo tabla CLIENTE');

Endif;

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('ins\_CLIENTE',

:OLD.dni,:NEW.dni,:NEW.usuario,sysdate);

Elsif deleting then

If :OLD.usuario IS NULL then

RAISE\_APPLICATION\_ERROR(-20003,'No se puede eliminar registro con usuario nulo Tabla CLIENTE');

Endif;

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('del\_PRODUCTO',

:OLD.dni,:NEW.dni,:OLD.usuario,sysdate);

End if;

exception

when others then

v\_msg:=sqlerrm;

raise\_application\_error(-20000,v\_msg);

END TR\_IN\_DE\_AC\_CLIENTE;

/

create or replace

trigger TR\_IN\_DE\_AC\_PEDIDO

AFTER INSERT OR DELETE OR UPDATE ON PEDIDO

FOR EACH ROW

DECLARE

v\_msgVARCHAR2(100);

BEGIN

if updating then --Cuando se Actualiza

if :NEW.usuario IS NULL then

RAISE\_APPLICATION\_ERROR(-20000,'No se puede insertar usuario nulo en tabla PEDIDO');

end if;

if updating('nropedido') then

RAISE\_APPLICATION\_ERROR(-20001, 'No se puede modificar atributo NROPEDIDO');

end if;

if updating('dni') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PEDIDO campo: dni',

:OLD.dni,:NEW.dni,:NEW.usuario,sysdate);

end if;

if updating('fecha') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PEDIDO campo: fecha',

:OLD.fecha,:NEW.fecha,:NEW.usuario,sysdate);

end if;

if updating('subtotalpedido') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PEDIDO campo: subtotalpedido',

:OLD.subtotalpedido,:NEW.subtotalpedido,:NEW.usuario,sysdate);

end if;

if updating('igv') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PEDIDO campo: igv',

:OLD.igv,:NEW.igv,:NEW.usuario,sysdate);

end if;

if updating('totalpedido') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PEDIDO campo: totalpedido',

:OLD.totalpedido,:NEW.totalpedido,:NEW.usuario,sysdate);

end if;

if updating('estado') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PEDIDO campo: estado',

:OLD.estado,:NEW.estado,:NEW.usuario,sysdate);

end if;

elsif inserting then

if :NEW.usuario IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20002,'No se puede insertar usuario nulo Tabla PEDIDO');

Endif;

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('ins\_PEDIDO',

:OLD.nropedido,:NEW.nropedido,:NEW.usuario,sysdate);

elsif deleting then

if :OLD.usuario IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20003,'No se puede eliminar registro con usuario nulo Tabla PEDIDO');

Endif;

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('del\_PEDIDO',

:OLD.nropedido,:NEW.nropedido,:OLD.usuario,sysdate);

End if;

exception

when others then

v\_msg:=sqlerrm;

raise\_application\_error(-20000,v\_msg);

END;

/

create or replace

trigger TR\_IN\_DE\_AC\_PRODUCTO

AFTER INSERT OR DELETE OR UPDATE ON PRODUCTO

FOR EACH ROW

DECLARE

v\_msgvarchar2(100);

BEGIN

if updating then

if :NEW.usuario IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20000,'No se puede insertar usuario nulo');

End if;

if updating('codproducto') then

RAISE\_APPLICATION\_ERROR(-20001,'No se puede modificar atributo CODPRODUCTO');

end if;

if updating('descrproducto') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PRODUCTO campo: descrproducto',

:OLD.descrproducto,:NEW.descrproducto,:NEW.usuario,sysdate);

end if;

if updating('preciounitario') then

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PRODUCTO campo: preciounitario',

:OLD.preciounitario,:NEW.preciounitario,:NEW.usuario,sysdate);

endif;

if updating('stock') then

insert into AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('upd\_PRODUCTO campo: stock',

:OLD.stock,:NEW.stock,:NEW.usuario,sysdate);

end if;

Elsif inserting then

if :NEW.usuario IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20002,'No se puede insertar usuario nulo Tabla PRODUCTO');

Endif;

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('ins\_PRODUCTO',

:OLD.codproducto,:NEW.codproducto,:NEW.usuario,sysdate);

Elsif deleting then

if :OLD.usuario IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20003,'No se puede eliminar registro con usuario nulo Tabla PRODUCTO');

Endif;

insertinto AUDITORIA(cambio,valor\_inicio,valor\_modif,usuario,fecha) values('del\_PRODUCTO',

:OLD.codproducto,:NEW.codproducto,:OLD.usuario,sysdate);

end if;

exception

when others then

v\_msg:=sqlerrm;

raise\_application\_error(-20000, v\_msg);

END TR\_IN\_DE\_AC\_PRODUCTO;

/

1. **Otros Bloques Utiles**

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

-- SECUENCIA

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

CREATE SEQUENCE AUDIT\_SEQUENCE

START WITH 1

INCREMENT BY 1;

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

-- TRIGGER

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

createorreplace

trigger TR\_SEQ\_AUDITORIA

BEFORE INSERT

ON AUDITORIA

REFERENCING NEW AS NEW

FOR EACH ROW

BEGIN

SELECT AUDIT\_SEQUENCE.nextvalINTO :NEW.CODSEGUIMIENTO FROM dual;

END;

/

1. **Implementación de requisitos**

**Requisito de diseño:**

**Registro de clientes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **Descripción de Funcionalidad** | | | | | |
| **Campos:** | | | | | |
|  | | Nombre | | Ingreso de los nombre del cliente. (Máximo 30, String) | |
|  | | DNI | | Ingreso del DNI. (Fijo 8, String) | |
|  | | Fecha Nacimiento | | Selección de Fecha de Nacimiento. | |
|  | | Direccion | | Ingreso del Domicilio (Máximo 80, String) | |
|  | | Email | | Ingreso del DNI del Cliente | |
|  | | Usuario | | Ingreso del nombre de usuario del Cliente | |
|  | | Contraseña | | Ingreso de la contraseña del cliente | |
| **Botones de Acción:** | | | | | |
|  | Grabar | | | Graba los datos ingresados | |
| **2** | **Flujo de Eventos** | | | | | |
|  | Flujo Básico | | | | 1. El actor ingresa al navegador de Internet y digita la siguiente dirección electrónica: <http://servidor:puerto/>RegistroClientes 2. El actor se dirige hacia el módulo de Registro 3. El actor pulsa el botón Registrar 4. El actor ingresa los datos necesarios. 5. El actor pulsa el botón Grabar | |
|  | Flujo Alternativo | | | | * Si no se llenan todos los campos, se mostrará un mensaje de error. | |
| **3** | **Consideraciones y/o Validaciones** | | | | | |
| Reglas de  Negocio | | | * El actor debe ingresar todos los campos. * El actor grabará los datos, los cuales se guardarán en una base de datos. * Se generará un código automáticamente al guardar el trabajador. | | |
| Validaciones | | | Comprobar | | Comprobar que no se dejen espacios en blanco |

**Registro de pedidos**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **Descripción de Funcionalidad** | | | | | |
| **Campos:** | | | | | |
|  | | DNI | | Ingreso del DNI. (Fijo 8, String) | |
|  | | fecha | | Fecha de Pedido | |
|  | |  | |  | |
|  | |  | |  | |
| **Botones de Acción:** | | | | | |
|  | Grabar | | | Graba los datos ingresados | |
| **2** | **Flujo de Eventos** | | | | | |
|  | Flujo Básico | | | | 1. El actor ingresa al navegador de Internet y digita la siguiente dirección electrónica: <http://servidor:puerto/Pedidos> 2. El actor se dirige hacia el módulo de Trabajador 3. El actor pulsa el botón Registrar 4. El actor ingresa los datos necesarios. 5. El actor pulsa el botón Grabar | |
|  | Flujo Alternativo | | | | * Si no se llenan todos los campos, se mostrará un mensaje de error. | |
| **3** | **Consideraciones y/o Validaciones** | | | | | |
| Reglas de  Negocio | | | * El actor debe ingresar todos los campos. * El actor grabará los datos, los cuales se guardarán en una base de datos. * Se generará un código automáticamente al guardar el pedido | | |
| Validaciones | | | Comprobar | | Comprobar que no se dejen espacios en blanco |

**Registro de Pagos**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **Descripción de Funcionalidad** | | | | | |
| **Campos:** | | | | | |
|  | | DNI | | Ingreso del DNI. (Fijo 8, String) | |
|  | | fecha | | Fecha del Pago | |
|  | |  | |  | |
|  | |  | |  | |
| **Botones de Acción:** | | | | | |
|  | Grabar | | | Graba los datos ingresados | |
| **2** | **Flujo de Eventos** | | | | | |
|  | Flujo Básico | | | | 1. El actor ingresa al navegador de Internet y digita la siguiente dirección electrónica: <http://servidor:puerto/Pagos> 2. El actor se dirige hacia el módulo de Trabajador 3. El actor pulsa el botón Registrar 4. El actor ingresa los datos necesarios. 5. El actor pulsa el botón Grabar | |
|  | Flujo Alternativo | | | | * Si no se llenan todos los campos, se mostrará un mensaje de error. | |
| **3** | **Consideraciones y/o Validaciones** | | | | | |
| Reglas de  Negocio | | | * El actor debe ingresar todos los campos. * El actor grabará los datos, los cuales se guardarán en una base de datos. * Se generará un código automáticamente al guardar el Pago | | |
| Validaciones | | | Comprobar | | Comprobar que no se dejen espacios en blanco |

**Login**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **Descripción de Funcionalidad** | | | | | |
| **Campos:** | | | | | |
|  | | Usuario | | Ingreso del Usuario (Fijo 50, String) | |
|  | | Contraseña | | Ingreso de la contraseña (Fijo 80, String) | |
|  | |  | |  | |
|  | |  | |  | |
| **Botones de Acción:** | | | | | |
|  | Grabar | | | Graba los datos ingresados | |
|  |  | Registrar | | | Abre el formulario para registrar datos | |
| **2** | **Flujo de Eventos** | | | | | |
|  | Flujo Básico | | | | 1. El actor ingresa al navegador de Internet y digita la siguiente dirección electrónica: <http://servidor:puerto/Login> 2. El actor se dirige hacia el módulo de Trabajador 3. El actor pulsa el botón Registrar 4. El actor ingresa los datos necesarios. 5. El actor pulsa el botón Grabar | |
|  | Flujo Alternativo | | | | * Si no se llenan todos los campos, se mostrará un mensaje de error. | |
| **3** | **Consideraciones y/o Validaciones** | | | | | |
| Reglas de  Negocio | | | * El actor debe ingresar todos los campos. * El actor grabará los datos, los cuales se guardarán en una base de datos. | | |
| Validaciones | | | Comprobar | | Comprobar que no se dejen espacios en blanco |

1. **Diagrama de Clases**

