Documento de análisis y diseño iteración 3

Cambios modelo conceptual

Cambios clase carrito

Desde la primera iteración, por las descripciones dadas en el documento de contexto del caso, habíamos visionado el uso de un carrito para las compras, por lo que al realizar los primeros modelos conceptual y relacional del proyecto, se incluyo la entidad (clase) y la tabla de carrito, sin embargo a estos no se les dio uso debido a que las indicaciones de la segunda iteración proponían una forma distinta a la inicialmente planteada para el pago de productos, por lo que el concepto de carrito era innecesario. Para esta tercera iteración se propuso finalmente el uso de un carrito para el proceso de compra, sin embargo las indicaciones a seguir que detallaban su uso nos obligaron a replantearnos la implementación.

Aquí podemos observar la implementación original en la base de datos:

```
CREATE TABLE Carrito
(IDCLIENTE NUMBER,
IDPRODUCTO NUMBER NOT NULL);

ALTER TABLE Carrito
ADD CONSTRAINT PK_CARRITO
PRIMARY KEY (IDCLIENTE,IDPRODUCTO);

ALTER TABLE Carrito
ADD CONSTRAINT FK_PROD_CARRITO
FOREIGN KEY (IDPRODUCTO) REFERENCES PRODUCTO (ID);

ALTER TABLE Carrito
ADD CONSTRAINT FK_CLIENTE_CARRITO
FOREIGN KEY (IDCLIENTE) REFERENCES CLIENTE (ID);

INSERT INTO Carrito VALUES (1, 3);
INSERT INTO Carrito VALUES (1, 5);
INSERT INTO Carrito VALUES (1, 6);
INSERT INTO Carrito VALUES (4, 10);
INSERT INTO Carrito VALUES (5, 7);
INSERT INTO Carrito VALUES (2, 2);
INSERT INTO Carrito VALUES (3, 1);
INSERT INTO Carrito VALUES (3, 2);
INSERT INTO Carrito VALUES (3, 4);
INSERT INTO Carrito VALUES (6, 11);
INSERT INTO Carrito VALUES (6, 9);
```

Con la descripción original pensamos que la mejor forma de implementar un carrito era asociando el id de un producto que cogía el cliente con su id. Sin embargo nos dimos cuenta que esta implementación causaba varias complicaciones, debido a que no se podía saber que cantidad del producto cogía el cliente, y para los requerimiento funcionales de esta iteración, no se podían

cumplir un numero de funciones: abandonar un carrito, coger un carrito vacio, devolver los productos en un carrito, etc.

Considerando lo anterior, decidimos que era mejor implementarlo de esta manera:

-Colocarle un id al carrito, para que esta tenga independencia del cliente y por lo tanto pueda ser abandonado, re asignado,etc.

-Dejar la tabla solo con el id del carrito y el id de su actual ocupante, si no se tiene ocupante se colocará el id del cliente en 0, que es la forma de representar que no pertenece a ningún cliente.

-Crear otra tabla llamada contiene carrito, la función de esta tabla es llevar el registro de los productos que se tienen en el carrito, las columnas de la tabla son el id del carrito, el id del producto y la cantidad que lleva de este producto, de esta forma un carrito puede tener varios productos al mismo tiempo y a si mismo puede haber una cantidad mayor a uno de un producto especifico.

La implementación nueva quedo así:

```
CREATE TABLE Carrito
(ID NUMBER NOT NULL,
IDCLIENTE NUMBER);

ALTER TABLE Carrito
ADD CONSTRAINT PK_CARRITO
PRIMARY KEY (ID);

ALTER TABLE Carrito
ADD CONSTRAINT FK_CLIENTE_CARRITO
FOREIGN KEY (IDCLIENTE) REFERENCES CLIENTE (ID);

--------------------------------Tabla Carrito_Contiene-----------------------------

CREATE TABLE CONTIENE_CARRITO
(IDCARRITO NUMBER NOT NULL,
IDPRODUCTO NUMBER NOT NULL,
CANTIDAD NUMBER NOT NULL);

ALTER TABLE CONTIENE_CARRITO
ADD CONSTRAINT PK_CONTIENE_CARRITO
PRIMARY KEY (IDCARRITO,IDPRODUCTO);

ALTER TABLE CONTIENE_CARRITO
ADD CONSTRAINT FK_CONTIENE_CARRITO
FOREIGN KEY (IDCARRITO) REFERENCES CARRITO (ID);

ALTER TABLE CONTIENE_CARRITO
ADD CONSTRAINT FK_PRODUCTO_CARRITO
FOREIGN KEY (IDPRODUCTO) REFERENCES PRODUCTO (ID);
```

Cambios pedido.

Para esta iteración se propuso cambiar la forma en la que se realizan los pedidos, para la iteración pasada, simplemente cuando un gerente de sucursal quería realizar un pedio o cuando se realizaba automáticamente debido a que se alcanzaba el nivel de reorden de un producto, solo tocaba esperar a la llegada del pedido para concretar el proceso. En esta nueva fase del proyecto se propuso un nuevo paso, consolidar los pedidos. Básicamente con este nuevo paso las ordenes de pedido que se hacían en la iteración anterior ahora tienen que agruparse por proveedor para una sucursal, de forma que el gerente tiene que utilizar la aplicación para consolidar los pedidos, es decir hacerlos realmente ya que las ordenes no se solicitarán al proveedor hasta que haya una orden de pedido consolidado.

Esta era la forma en la que se hacia previamente a esta iteración:

```
CREATE TABLE Pedido
(IDPEDIDO NUMBER,
ID_PROVEEDOR NUMBER,
ID_PRODUCTO NUMBER NOT NULL,
VOLUMEN NUMBER NOT NULL,
PRECIO_TOTAL NUMBER NOT NULL,
FECHA_ESPERADA DATE NOT NULL,
FECHA_ENTREGA DATE,
CALIFICACION NUMBER (2,1),
ESTADO VARCHAR2(255) NOT NULL,
SUCURSAL NUMBER NOT NULL);

ALTER TABLE Pedido
ADD CONSTRAINT PK_IDPEDIDO
PRIMARY KEY (IDPEDIDO);

ALTER TABLE Pedido
ADD CONSTRAINT FK_ID_PRODUCTO
FOREIGN KEY (ID_PRODUCTO) REFERENCES Producto(ID);

ALTER TABLE Pedido
ADD CONSTRAINT FK_PROVEEDOR
FOREIGN KEY (ID_PROVEEDOR) REFERENCES Proveedor(ID);

ALTER TABLE Pedido
ADD CONSTRAINT FK_SUCURSAL_PEDIDO
FOREIGN KEY (SUCURSAL) REFERENCES Sucursal(ID);

ALTER TABLE Pedido
ADD CONSTRAINT CK_PRECIO_TOTAL
CHECK (PRECIO_TOTAL > 0);

ALTER TABLE Pedido
ADD CONSTRAINT CK_VOLUMEN_PED
CHECK (VOLUMEN > 0);
```

Considerando el análisis hecho previamente decidimos que para poder construir el nuevo proceso de pedio propuesto era necesario mantener la tabla de pedidos pero agregando una nueva tabla para los pedidos consolidados.

Debido a que las ordenes de pedido se generan antes de hacer un pedido consolidado, realmente no tienen varias de las características que se les había asginado originalmente, como fecha esperada, fecha de llegada, estado y calificación, estas ahora pertenecerán a el pedido consolidado, debido a que este es el verdadero pedido que se le realiza al proveedor.

Realizamos puntualmente los siguientes cambios en la tabla de pedido:

-Va a tener un id de pedido que va a corresponder al pedido consolidado al que pertenece, como las ordenes de pedido se crean antes de que se consolide el pedido, al crearse la orden se le va a asignar el id de pedido 0, que no corresponde a ningún pedido consolidado, el id se va a cambiar una vez esta orden haga parta de un pedido consolidado. Es importante que se asocie una orden de pedido con un pedido consolidado ya que una vez llegue el pedido, se tiene que saber que se incluía en el pedido consolidado.

-Adicionalmente a lo anterior se mantienen las columnas de idproveedor, idproducto, volumen, precio sub total y sucursal. Todos estos datos son necesarios para la que se pueda realizar los procesos de consolidar pedio y llegada de pedido consolidado.

Las nuevas tablas quedan asi

Pedido

```
----------------------Tabla Pedido---------------------------------
CREATE TABLE Pedido
(IDPEDIDO NUMBER,
IDPROVEEDOR NUMBER,
IDPRODUCTO NUMBER NOT NULL,
VOLUMEN NUMBER NOT NULL,
PRECIOSUBTOTAL NUMBER NOT NULL,
SUCURSAL NUMBER NOT NULL);

ALTER TABLE Pedido
ADD CONSTRAINT PK_IDPEDIDO
PRIMARY KEY (IDPROVEEDOR,IDPRODUCTO,VOLUMEN,SUCURSAL);

ALTER TABLE Pedido
ADD CONSTRAINT FK_IDPRODUCTO
FOREIGN KEY (IDPRODUCTO) REFERENCES Producto(ID);

ALTER TABLE Pedido
ADD CONSTRAINT FK_PROVEEDOR
FOREIGN KEY (IDPROVEEDOR) REFERENCES Proveedor(ID);

ALTER TABLE Pedido
ADD CONSTRAINT FK_SUCURSAL_PEDIDO
FOREIGN KEY (SUCURSAL) REFERENCES Sucursal(ID);

ALTER TABLE Pedido
ADD CONSTRAINT FK_PEDIDO_CON
FOREIGN KEY (IDPEDIDO) REFERENCES PEDIDO_CONSOLIDADO(IDPEDIDO);

ALTER TABLE Pedido
ADD CONSTRAINT CK_VOLUMEN_PED
CHECK (VOLUMEN > 0);
```

Pedido consolidado

```
CREATE TABLE PEDIDO_CONSOLIDADO
(IDPEDIDO NUMBER,
IDPROVEEDOR NUMBER,
PRECIOTOTAL NUMBER NOT NULL,
FECHAESPERADA TIMESTAMP NOT NULL,
FECHAENTREGA TIMESTAMP,
CALIFICACION NUMBER (2,1),
ESTADO VARCHAR2(255) NOT NULL,
SUCURSAL NUMBER NOT NULL);

ALTER TABLE PEDIDO_CONSOLIDADO
ADD CONSTRAINT PK_IDPEDIDO_CONSOLIDADO
PRIMARY KEY (IDPEDIDO);

ALTER TABLE PEDIDO_CONSOLIDADO
ADD CONSTRAINT FK_PROVEEDOR_PEDIDO_CONSOLIDADO
FOREIGN KEY (IDPROVEEDOR) REFERENCES Proveedor(ID);

ALTER TABLE PEDIDO_CONSOLIDADO
ADD CONSTRAINT FK_SUCURSAL_PEDIDO_CONSOLIDADO
FOREIGN KEY (SUCURSAL) REFERENCES Sucursal(ID);

ALTER TABLE PEDIDO_CONSOLIDADO
ADD CONSTRAINT CK_PRECIOTOTAL_PEDIDO_CONSOLIDADO
CHECK (PRECIOTOTAL > 0);

ALTER TABLE PEDIDO_CONSOLIDADO
ADD CONSTRAINT CK_ESTADO_PEDIDO
CHECK (ESTADO IN ('Entregado', 'Pendiente'));
```

Listado de tablas con sus metadatos

	NOMBRETABLA	NOMBRECOLUMNA	TIPODEDATO	NOMBRERESTRICCION	PERMITENULOS
1	BODEGA	CATEGORIAALMAC	NUMBER	FK CATEGORIA ALMACENAMIENTO	NO
2	BODEGA	CATEGORIAALMAC	NUMBER	SYS C00798997	NO
3	BODEGA	ID	NUMBER	PK BODEGA	NO
4	BODEGA	IDSUCURSAL	NUMBER	FK BODEGA SUC	NO
5	BODEGA	IDSUCURSAL	NUMBER	SYS C00798994	NO
6	BODEGA	PESOMAX	NUMBER	CK PESOMAX	NO
7	BODEGA	PESOMAX	NUMBER	SYS C00798996	NO
8	BODEGA	VOLUMENMAX	NUMBER	CK VOLMAX	NO
9	BODEGA	VOLUMENMAX	NUMBER	SYS C00798995	NO
10	CARRITO	ID	NUMBER	PK CARRITO	NO
11	CARRITO	ID	NUMBER	SYS C00814761	NO
12	CARRITO	IDCLIENTE	NUMBER	FK CLIENTE CARRITO	SI
13	CATEGORIA	ID	NUMBER	PK CATEGORIA	NO
14	CATEGORIA	NOMBRE	VARCHAR2	CK CATEGORIA	NO
15	CATEGORIA	NOMBRE	VARCHAR2	SYS C00798929	NO
16	CLIENTE	CORREO	VARCHAR2	SYS C00798982	NO
17	CLIENTE	DIRECCION	VARCHAR2	NO TIENE	SI
18	CLIENTE	ID	NUMBER	PK CLIENTE	NO
19	CLIENTE	NOMBRE	VARCHAR2	SYS C00798981	NO
20	CLIENTE	NUMERODOC	NUMBER	SYS C00798984	NO
21	CLIENTE	NUMERODOC	NUMBER	SYS C00798987	NO
22	CLIENTE	PUNTOS	NUMBER	SYS C00798985	NO
23	CLIENTE	TIPOCLIENTE	VARCHAR2	CK TIPOCLIENTE	NO
24	CLIENTE	TIPOCLIENTE	VARCHAR2	SYS C00798986	NO
25	CLIENTE	TIPODOC	VARCHAR2	SYS C00798983	NO
26	CONTIENE BODEGA	CANTIDAD	NUMBER	NO TIENE	SI
27	CONTIENE BODEGA	IDBODEGA	NUMBER	FK IDBODEGA CONTIENE BODEGA	NO
28	CONTIENE BODEGA	IDBODEGA	NUMBER	SYS C00799022	NO
29	CONTIENE BODEGA	IDPRODUCTO	NUMBER	FK IDPRODUCTO CONTIENE BODEGA	NO
30	CONTIENE BODEGA	IDPRODUCTO	NUMBER	PK CONTIENE BODEGA	NO
31	CONTIENE BODEGA	IDPRODUCTO	NUMBER	SYS C00799021	NO
32	CONTIENE CARRITO	CANTIDAD	NUMBER	SYS C00817632	NO
33	CONTIENE CARRITO	IDCARRITO	NUMBER	FK CONTIENE CARRITO	NO
34	CONTIENE CARRITO	IDCARRITO	NUMBER	PK CONTIENE CARRITO	NO
35	CONTIENE CARRITO	IDCARRITO	NUMBER	SYS C00817630	NO
36	CONTIENE CARRITO	IDPRODUCTO	NUMBER	FK PRODUCTO CARRITO	NO

	NOMBRETABLA	NOMBRECOLUMNA	TIPODEDATO	NOMBRERESTRICCION	PERMITENULOS
37	CONTIENE CARRITO	IDPRODUCTO	NUMBER	PK CONTIENE CARRITO	NO
38	CONTIENE CARRITO	IDPRODUCTO	NUMBER	SYS C00817631	NO
39	CONTIENE ESTANTE	CANTIDAD	NUMBER	NO TIENE	SI
40	CONTIENE ESTANTE	IDESTANTE	NUMBER	FK ESTANTE CONTIENE ESTANTE	NO
41	CONTIENE ESTANTE	IDESTANTE	NUMBER	SYS C00799027	NO
42	CONTIENE ESTANTE	IDPRODUCTO	NUMBER	FK IDPRODUCTO CONTIENE ESTANTE	NO
43	CONTIENE ESTANTE	IDPRODUCTO	NUMBER	PK CONTIENE ESTANTE	NO
44	CONTIENE ESTANTE	IDPRODUCTO	NUMBER	SYS C00799026	NO
45	ESTANTE	CATEGORIAALMAC	NUMBER	FK E CATEGORIAALMACENAMIENTO	NO
46	ESTANTE	CATEGORIAALMAC	NUMBER	SYS C00799007	NO
47	ESTANTE	ID	NUMBER	PK ESTANTE	NO
48	ESTANTE	IDSUCURSAL	NUMBER	FK ESTANTE SUC	NO
49	ESTANTE	IDSUCURSAL	NUMBER	SYS C00799003	NO
50	ESTANTE	NIVELABASTECIMIENTO	NUMBER	CK NV ABASTECIMIENTO	NO
51	ESTANTE	NIVELABASTECIMIENTO	NUMBER	SYS C00799004	NO
52	ESTANTE	PESOMAX	NUMBER	CK E PESOMAX	NO
53	ESTANTE	PESOMAX	NUMBER	SYS C00799006	NO
54	ESTANTE	VOLUMENMAX	NUMBER	CK E VOLMAX	NO
55	ESTANTE	VOLUMENMAX	NUMBER	SYS C00799005	NO
56	PEDIDO	IDPEDIDO	NUMBER	FK PEDIDO PEDIDOCON	SI
57	PEDIDO	IDPRODUCTO	NUMBER	FK IDPRODUCTO	NO
58	PEDIDO	IDPRODUCTO	NUMBER	PK IDPEDIDO	NO
59	PEDIDO	IDPRODUCTO	NUMBER	SYS C00819654	NO
60	PEDIDO	IDPROVEEDOR	NUMBER	FK PROVEEDOR	NO
61	PEDIDO	IDPROVEEDOR	NUMBER	PK IDPEDIDO	NO
62	PEDIDO	PRECIOSUBTOTAL	NUMBER	SYS C00819656	NO
63	PEDIDO	SUCURSAL	NUMBER	FK SUCURSAL PEDIDO	NO
64	PEDIDO	SUCURSAL	NUMBER	PK IDPEDIDO	NO
65	PEDIDO	SUCURSAL	NUMBER	SYS C00819657	NO
66	PEDIDO	VOLUMEN	NUMBER	CK VOLUMEN PED	NO
67	PEDIDO	VOLUMEN	NUMBER	PK IDPEDIDO	NO
68	PEDIDO	VOLUMEN	NUMBER	SYS C00819655	NO
69	PEDIDO CONSOL...	CALIFICACION	NUMBER	NO TIENE	SI
70	PEDIDO CONSOL...	ESTADO	VARCHAR2	CK ESTADO PEDIDO	NO
71	PEDIDO CONSOL...	ESTADO	VARCHAR2	SYS C00819647	NO
72	PEDIDO CONSOL...	FECHAENTREGA	TIMEST...	NO TIENE	SI

	NOMBRETABLA	NOMBRECOLUMNA	TIPODEDATO	NOMBRERESTRICCION	PERMITENULOS
73	PEDIDO CONSOL...	FECHAESPERADA	TIMEST...	SYS C00819646	NO
74	PEDIDO CONSOL...	IDPEDIDO	NUMBER	PK IDPEDIDO CONSOLIDADO	NO
75	PEDIDO CONSOL...	IDPROVEEDOR	NUMBER	FK PROVEEDOR PEDIDO CONSOLI...	SI
76	PEDIDO CONSOL...	PRECIOTOTAL	NUMBER	CK PRECIOTOTAL PEDIDO CONSO...	NO
77	PEDIDO CONSOL...	PRECIOTOTAL	NUMBER	SYS C00819645	NO
78	PEDIDO CONSOL...	SUCURSAL	NUMBER	FK SUCURSAL PEDIDO CONSOLIDADO	NO
79	PEDIDO CONSOL...	SUCURSAL	NUMBER	SYS C00819648	NO
80	PRODUCTO	CANTIDAD	NUMBER	SYS C00798936	NO
81	PRODUCTO	CODIGODEBARRAS	VARCHAR2	SYS C00798942	NO
82	PRODUCTO	FECHAVENCIMIENTO	TIMEST...	NO TIENE	SI
83	PRODUCTO	ID	NUMBER	PK IDPRODUCTO	NO
84	PRODUCTO	IDCATEGORIA	NUMBER	FK CATPRODUCTO	NO
85	PRODUCTO	IDCATEGORIA	NUMBER	SYS C00798933	NO
86	PRODUCTO	MARCA	VARCHAR2	SYS C00798934	NO
87	PRODUCTO	NIVELREORDEN	NUMBER	SYS C00798943	NO
88	PRODUCTO	NOMBRE	VARCHAR2	SYS C00798932	NO
89	PRODUCTO	PESO	NUMBER	CK PESO	NO
90	PRODUCTO	PESO	NUMBER	SYS C00798940	NO
91	PRODUCTO	PRECIOUNITARIO	NUMBER	CK PRECIOUNITARIO	NO
92	PRODUCTO	PRECIOUNITARIO	NUMBER	SYS C00798935	NO
93	PRODUCTO	PRECIOXUNDMEDIDA	NUMBER	SYS C00798938	NO
94	PRODUCTO	PRESENTACION	VARCHAR2	SYS C00798939	NO
95	PRODUCTO	SUBCATEGORIA	VARCHAR2	SYS C00798944	NO
96	PRODUCTO	UNIDADMEDIDA	VARCHAR2	CK UNIDADMEDIDA	NO
97	PRODUCTO	UNIDADMEDIDA	VARCHAR2	SYS C00798937	NO
98	PRODUCTO	VOLUMEN	NUMBER	SYS C00798941	NO
99	PRODUCTO PROV...	CALIFICACION	NUMBER	CK CALIFICACION	NO
100	PRODUCTO PROV...	CALIFICACION	NUMBER	SYS C00798953	NO
101	PRODUCTO PROV...	IDPRODUCTO	NUMBER	FK IDPRODUCTO PROV	NO
102	PRODUCTO PROV...	IDPRODUCTO	NUMBER	PK IDPRODUCTO PROV	NO
103	PRODUCTO PROV...	IDPROVEEDOR	NUMBER	FK IDPROVEEDOR PROD	NO
104	PRODUCTO PROV...	IDPROVEEDOR	NUMBER	PK IDPRODUCTO PROV	NO
105	PRODUCTO PROV...	IDPROVEEDOR	NUMBER	SYS C00798950	NO
106	PRODUCTO PROV...	NOMBRE	VARCHAR2	SYS C00798951	NO
107	PRODUCTO PROV...	PRECIO	NUMBER	CK PRECIO	NO
108	PRODUCTO PROV...	PRECIO	NUMBER	SYS C00798952	NO

	NOMBRETABLA	NOMBRECOLUMNA	TIPODEDATO	NOMBRERESTRICCION	PERMITENULOS
109	PROMOCION	CANTIDAD	NUMBER	NO TIENE	SI
110	PROMOCION	FECHAFIN	TIMEST...	CK PROMOCION FECHA	NO
111	PROMOCION	FECHAFIN	TIMEST...	SYS C00798976	NO
112	PROMOCION	FECHAINICIO	TIMEST...	CK PROMOCION FECHA	NO
113	PROMOCION	FECHAINICIO	TIMEST...	SYS C00798975	NO
114	PROMOCION	IDPRODUCTO	NUMBER	FK PROMOCION	NO
115	PROMOCION	IDPRODUCTO	NUMBER	SYS C00798972	NO
116	PROMOCION	IDPROMOCION	NUMBER	PK PROMOCION	NO
117	PROMOCION	IDSUCURSAL	NUMBER	FK SUC PROMOCION	NO
118	PROMOCION	IDSUCURSAL	NUMBER	SYS C00798973	NO
119	PROMOCION	TIPO	VARCHAR2	SYS C00798974	NO
120	PROVEEDOR	ID	NUMBER	PK IDPROVEEDOR	NO
121	PROVEEDOR	NIT	NUMBER	SYS C00798924	NO
122	PROVEEDOR	NOMBRE	VARCHAR2	SYS C00798925	NO
123	PROVEEDOR	SUCURSAL	NUMBER	FK SUCURSAL PROVEEDOR	NO
124	PROVEEDOR	SUCURSAL	NUMBER	SYS C00798926	NO
125	RESERVAS	ESTADO	VARCHAR2	CK R ESTADO	SI
126	RESERVAS	FECHA	DATE	NO TIENE	SI
127	RESERVAS	IDCLIENTE	NUMBER	NO TIENE	SI
128	RESERVAS	IDFUNCION	NUMBER	NO TIENE	SI
129	RESERVAS	IDRESERVA	NUMBER	PK RESERVAS	NO
130	ROL USUARIO	ID	NUMBER	PK ROL USUARIO	NO
131	ROL USUARIO	NOMBRE	VARCHAR2	SYS C00798908	NO
132	SILLASRESERVAS	IDFUNCION	NUMBER	PK SILLASRESERVAS	NO
133	SILLASRESERVAS	IDRESERVA	NUMBER	FK SR RESERVAS	NO
134	SILLASRESERVAS	IDRESERVA	NUMBER	SYS C00810140	NO
135	SILLASRESERVAS	IDSILLA	VARCHAR2	PK SILLASRESERVAS	NO
136	SUCURSAL	CIUDAD	VARCHAR2	SYS C00798910	NO
137	SUCURSAL	DIRECCION	VARCHAR2	SYS C00798911	NO
138	SUCURSAL	ID	NUMBER	PK SUCURSAL	NO
139	SUCURSAL	NOMBRE	VARCHAR2	SYS C00798912	NO
140	USUARIO	CONTRASENA	VARCHAR2	SYS C00798916	NO
141	USUARIO	CORREO	VARCHAR2	SYS C00798915	NO
142	USUARIO	ID	NUMBER	PK USUARIO	NO
143	USUARIO	NOMBRE	VARCHAR2	SYS C00798914	NO
144	USUARIO	NUMERODOC	NUMBER	SYS C00798918	NO
145	USUARIO	NUMERODOC	NUMBER	SYS C00798920	NO
146	USUARIO	SUCURSAL	NUMBER	FK SUCURSAL	SI
147	USUARIO	TIPODOC	VARCHAR2	SYS C00798917	NO
148	USUARIO	TIPOUSUARIO	NUMBER	FK TIPOUSUARIO	NO
149	USUARIO	TIPOUSUARIO	NUMBER	SYS C00798919	NO
150	VENTA	CANTIDAD	NUMBER	NO TIENE	SI
151	VENTA	FECHAVENTA	TIMEST...	SYS C00799014	NO
152	VENTA	ID	NUMBER	PK VENTA	NO
153	VENTA	IDCLIENTE	NUMBER	FK ID CLIENTE VENTA	SI
154	VENTA	IDPRODUCTO	NUMBER	FK ID PRODUCTO VENTA	NO
155	VENTA	IDPRODUCTO	NUMBER	PK VENTA	NO
156	VENTA	SUCURSAL	NUMBER	FK SUCURSAL VENTA	SI
157	VENTA	TOTAL	NUMBER	CK TOTAL VENTA	NO
158	VENTA	TOTAL	NUMBER	SYS C00799015	NO

Documentación lógica aplicación

RF15

La implementación de este requerimiento empieza en la interfaz, donde al ejecutarse le pedirá al usuario (cliente) su correo, el input ingresado por el usuario se pasara a la función solicitarCarrito de la clase de lógica de superAndes

```
public void solicitarCarrito( )
{
    try
    {
            String correo =  JOptionPane.showInputDialog (this, "Ingrese su correo ", "Adicionar correo", JOptionPane.QUESTION_MESSAGE);

            long idcarrito = superAndes.solicitarCarrito(correo);
```

Una vez en el método se llamara a el método de la persistencia darClientePorCorreo(), pasando por parámetro el correo obtenido en la interfaz, este método devolverá un objeto de la clase cliente, que tendrá toda la información del cliente que esta solicitando el carrito. Se llamara al método getId() del objeto para conseguir el id del cliente. Se llamara al método de la persistencia darCarritosDisponibles() que devolverá todos los carritos existentes que no tengan asignado un cliente, el resultado del método se guardara en una lista de objetos de la clase carrito,si no existe ningún carrito libre se devolverá una lista vacia.

```
public long solicitarCarrito(String correo) {
    Cliente cliente = ps.darClientePorCorreo(correo);
    long idcliente= cliente.getId();

    List<Carrito> carritos= ps.darCarritosDisponibles();
```

Posteriormente con un if se evaluara si la lista esta vacia (lo que quiere decir que no hay carritos libres), en caso de cumplirse la condición se llamara al método de la persistencia adicionarCarrito(), que creara un nuevo carrito en la base de datos sin asignarle un cliente, el método devolverá un objeto de la clase carrito. Se llamara al método getId del objeto para obtener el id del carrito que acabamos de crear , posteriormente se llamara al método de la persistencia asginarCarrito() pasando por parámetro el id del carrito creado y el id del cliente, el metododo hará cambios en la base de datos para que el carrito creado se encuentre asignado al cliente que lo esta solicitando.

```
if(carritos.isEmpty()) {

      Carrito carrito= ps.adicionarCarrito();
      long idcarrito= carrito.getId();
      ps.asignarCarrito(idcarrito, idcliente);
      return idcarrito;
  }
```

En caso de que la condición del if no se cumpla, quiere decir que si hay carritos disponibles, por lo tanto se llamara al método get de la lista de carritos, pasando por parámetro el numero 0 que

hará que se obtenga el primer carrito de la lista, llamando al método get id del carrito obtenemos su identificador y llamamos al método limpiarCarrito pasándolo por parámetro, este método va a realizar la función de devolver los productos a los estantes si el carrito disponible aun tiene estos. Finalmente se llamara a la función asignar carrito con el id del cliente y el id del carrito.

```
else {

    Carrito carrito = carritos.get(0);
    long idcarrito= carrito.getId();
    limpiarCarrito(idcarrito);
    ps.asignarCarrito(idcarrito, idcliente);
    return idcarrito;

}
```

RF16

La implementación empieza en la interfaz con el método adicionarProductoCarrito(), el método pide en la interfaz el correo, el código de barras del producto y la cantidad que se va a agregar al carrito, si estos datos no son nulos se va a llamar a el método del mismo nombre en la clase de lógica de la aplicación, pasando los 3 datos pedidos por parámetro

```
public void adicionarProductoCarrito( )
{
    try
    {

        String correo = JOptionPane.showInputDialog (this, "Ingrese su correo ", "Adicionar", JOptionPane.QUESTION_MESSAGE);
        String codigobarras = JOptionPane.showInputDialog (this, "Ingrese el codigo de barras del producto", "Adicionar ", JOptionPane.QUESTION_MES
        String cantidad = JOptionPane.showInputDialog (this, "Cantidad del producto que va a agregar", "Adicionar ", JOptionPane.QUESTION_MESSAGE);

        if (correo != null && codigobarras != null && cantidad != null )
        {
            superAndes.adicionarProductoCarrito(correo,codigobarras,Long.parseLong(cantidad));
```

En el método de la clase de lógica se va a obtener el id del cliente llamando al método de la persistencia darClientePorCorreo() y llamando el método getId sobre este método. Con el id del cliente se va a llamar al método de la persistencia darCarritoCliente() pasandolo por parámetro, este método nos va a devolver el objeto Carrito que correpondera al carrito que esta utilizando el cliente, se llama al método get id sobre el resultado para obtener el id del carrito. Con el código de barras obtenido por parámetro se llama al método de la persistencia darProductoPorCodigoBarras() y se obtiene el id de este. A continuación se llama al método de la persistencia actualizarContieneEstante() pasando por parámetro la cantidad obetenida por parámetro de manera negtiva y el id del producto, esto hará que el método quite la cantidad cogida del producto del estante donde se encuentra.

```
public void adicionarProductoCarrito(String correo, String codigobarras,long cantidad) {
    long idcliente =ps.darClientePorCorreo(correo).getId();
    long idcarrito= ps.darCarritoCliente(idcliente).getId();
    long idproducto = ps.darProductoPorCodigoBarras(codigobarras).getId();
    ps.actualizarContieneEstante(-cantidad, idproducto);
```

Posteriormente se llamara al método tieneProductoCarrito() de la persistencia pasando por parámetro el id del carrito y el id del producto, este método nos devolverá la entrada en la tabla de datos donde esta el carrito utilizado y el producto que se va agregar, si el método devuelvo el objeto nulo quiere decir que en el carrito no se ha agregado el producto anteriormente, si devuelve un objeto ContieneCarrito quiere decir que el producto ya se encuentra en el carrito entonces solo se debe aumentar la cantidad. Lo anterior se evalua con un if y se ejecutan una de las 2 opciones dependiendo del resultado.

```
ContieneCarrito cc=ps.tieneProductoCarrito(idcarrito,idproducto);

if(cc==null) {

    long res = ps.adicionarProductoCarrito(idcarrito,idproducto,cantidad);
    if(res!=-1) {
        log.info ("Se adiciono el producto al carrito");
    }
    else {
        log.info ("No se pudo adicionar producto");
    }
}
else {

    long res = ps.incrementarProductoCarrito(idcarrito,idproducto,cantidad);
    if(res!=-1) {
        log.info ("Se adiciono el producto al carrito");
    }
    else {
        log.info ("No se pudo adicionar producto");
    }
}
```

RF17

En la interfaz se empieza pidiendo por input al usuario su correo, el código de barras del producto que desea quitar del carrito y la cantidad del producto que desea quitar, se aclara que si se desea quitar todas las unidades del producto que se encuentran en el carro se coloca 0 cuando se pregunta la cantidad. Posteriormente se llama a la función de la clase de lógica quitarProductoCarrito con los datos pedidos.

```
public void quitarProductoCarrito( )
{
    try
    {

        String correo = JOptionPane.showInputDialog (this, "Ingrese su correo ", "Adicionar", JOptionPane.QUESTION_MESSAGE);
        String codigobarras = JOptionPane.showInputDialog (this, "Ingrese el codigo de barras del producto", "Adicionar ", JOptionPane.QUESTION_MESSAGE
        String cantidad = JOptionPane.showInputDialog (this, "Unidades del producto que desea quitar (Si desea quitar todas las unidades del producto i

        if (correo != null && codigobarras != null && cantidad != null )
        {
            superAndes.quitarProductoCarrito(correo,codigobarras,Long.parseLong(cantidad));
```

En el método en la lógica se llama al método de la persistencia darClientePorCorreo y a getId de este resultado para obtener el id del cliente, se obtiene el id del carrito del cliente por medio del metodo darCarritoCliente(idcliente).getId() y el id del producto llamando al método darProductoPorCodigoBarras(codigobarras).getId(). A continuación, se analiza si se paso un 0

como cantidad para eliminar todos los productos, esto por medio de un if. Si la cantidad si corresponde a 0 se llamara al método de la persistencia tieneProductoCarrito que nos devolverá la cantidad del producto que se encuentra en el carrito, se llamara al método quitarProductoCarrito, con el id del carrito y el id del producto, esto alterara la base de datos para que elimine el producto del carrito, siguiente a esto se llamara al método actualizarContieneEstante con la cantidad obtenida del producto y el id del producto, esto hará que en la base de datos se refleje que al estante del producto devuelto se devolvieron todas las unidades que se encontraban en el carrito. Si la cantidad es diferente de 0 quiere decir que se va a eliminar una cantidad especifica, se va a entrar al else del if y se llamara al método incrementarProductoCarrito, pasando el id del carrito el id del producto y la cantidad de forma negativa, lo que hará que en la base de datos se refleje la disminución en la cantidad de unidades que se encuentra en el carrito del producto. Finalmente se llamara a la función actualizarContieneEstante.

```
public void quitarProductoCarrito(String correo, String codigobarras, long cantidad) {
    long idcliente =ps.darClientePorCorreo(correo).getId();
    long idcarrito= ps.darCarritoCliente(idcliente).getId();
    long idproducto = ps.darProductoPorCodigoBarras(codigobarras).getId();
    if(cantidad==0) {
        long can=ps.tieneProductoCarrito(idcarrito, idproducto).getCantidad();
        ps.quitarProductoCarrito(idcarrito, idproducto);
        ps.actualizarContieneEstante(can, idproducto);

    }
    else {
        ps.incrementarProductoCarrito(idcarrito, idproducto, -cantidad);
        ps.actualizarContieneEstante(cantidad, idproducto);
    }
}
```

RF18

La implementación empieza en la interfaz con el método pagar(), donde se pide al usuario por input el correo del cliente que realiza la compra y la fecha actual. Los datos se pasan por parámetro en la llamada al método del mismo nombre en la clase de lógica.

```
public void pagar( )
{
    try
    {


        String cocliente = JOptionPane.showInputDialog (this, "Ingrese su correo", "Adicionar correo", JOptionPane.QUESTION_MESSAGE);
        String fecha = JOptionPane.showInputDialog (this, "Fecha de la compra? (AAAA-MM-DD)?", "Adicionar fecha", JOptionPane.QUESTION_MESSAGE);



        Timestamp fechae = java.sql.Timestamp.valueOf(fecha+" 00:00:00.000");


        if ( cocliente != null && fecha != null )
        {
            String p = superAndes.pagar (cocliente, fechae);
```

En el método de lógica se inicia la variable preciototal en 0, en esta se va a llevar la cuenta final a pagar por el cliente, también se inicializa la variable factura para dar el inicio al mensaje de factura que se devolverá, se inicia también la variable idventa en 0. Con el correo pasado por parámetro se obtiene el id del cliente al llamar a darClientePorCorreo(correo).getId(), y se obtiene el id del carrito con ps.darCarritoCliente(idcliente).getId(), con el id del carrito se van a obtener todos los productos que se encuentran en este llamando al método de la persistencia darProductosCarrito.

Que nos va a devolver una lista de objetos ContieneCarrito, donde cada objeto tiene el id del producto y la cantidad que se encuentra en el carrito.

```
public String pagar(String correo, Timestamp fecha) {
    double preciototal = 0;
    String factura= "FACTURA SUPERANDES\n-----------------------------------\nFecha: "+fecha;
    long idventa=0;

    long idcliente=ps.darClientePorCorreo(correo).getId();
    long idcarrito=ps.darCarritoCliente(idcliente).getId();
    List<ContieneCarrito> cc =ps.darProductosCarrito(idcarrito);
```

A continuacion se va a iterar sobre la lista con los productos del carrito, se obtiene el Objeto ContieneCarrito y se sacan los valores de la cantidad y el id del producto, se llama al método

ps.darProductoPorId(idproducto).getCantidad() que nos va a dar la cantidad actual del producto, vamos a llamar a ps.darProductoPorId(idproducto) para obtener el objeto producto si la cantidad que se va a comprar es mayor a la cantidad que se tiene del producto no se va a poder ejecutar la venta del producto por falta de inventario. Si por el contrario la cantidad es menor a la cantidad actual se va a actualizar el inventario del producto llamando a ps.actualizarProducto(-cantidad, idproducto), posteriormente se van a conseguir los datos de bodega del producto para ver si es necesario ejecutar una orden de pedido por el nivel de reorden.

```
public String pagar(String correo, Timestamp fecha) {
    double preciototal = 0;
    String factura= "FACTURA SUPERANDES\n-----------------------------------\nFecha: "+fecha;
    long idventa=0;

    long idcliente=ps.darClientePorCorreo(correo).getId();
    long idcarrito=ps.darCarritoCliente(idcliente).getId();
    List<ContieneCarrito> cc =ps.darProductosCarrito(idcarrito);

    for(int i=0;i<cc.size();i++) {
        ContieneCarrito contiene =cc.get(i);
        long cantidad = contiene.getCantidad();
        long idproducto= contiene.getIdproducto();
        long cantidadactual= ps.darProductoPorId(idproducto).getCantidad();
        Producto producto = ps.darProductoPorId(idproducto);
        if(cantidad<cantidadactual) {
            ps.actualizarProducto(-cantidad, idproducto);

            ContieneBodega infobodega = ps.darContieneBodega(idproducto);
            long idbodega= infobodega.getIdbodega();
            Bodega bodega= ps.darBodega(idbodega);
            long idsucursal= bodega.getIdsucursal();
            double volmax= bodega.getVolumenMax();
            double volactual= (cantidadactual-cantidad)*producto.getVolumen();
            double volpedido= volmax-volactual;
            ProductoProveedor pproveedor = ps.darProductoProveedor(idproducto);
            long idproveedor= pproveedor.getIdproveedor();
            double stpedido = (volpedido/producto.getVolumen())*pproveedor.getPrecio();
```

Si el nivel de reorden es mayor a la cantidad actual de l producto se llama a el método adicionarPedido() con los datos correspondientes para hacer la orden de pedido. Posteriormente se analiza si existen promociones existentes para el producto y se aplican los respectivos descuentos en caso de que se tengan.

```
if((cantidadactual-cantidad)<= producto.getNivelReorden()) {
    adicionarPedido(idproveedor,idproducto,volpedido,stpedido,idsucursal);
}

double totalproducto= producto.getPrecioUnitario()*cantidad;
Promocion promocion = ps.darPromocionPorProducto(idproducto);

    if(promocion != null && promocion.getCantidad()>0) {
        String tipo = promocion.getTipo();
        if(tipo.contains("descuento")) {
            int descuento = Integer.parseInt(tipo.substring( tipo.lastIndexOf("o")+2, tipo.indexOf("%") ));
            totalproducto -= ((totalproducto*descuento)/100);
            long idpromo = promocion.getId();
            actualizarPromocion((int)-cantidad ,idpromo);
        }
        else if(tipo.contains("pague")){
            int pague = Integer.parseInt(tipo.substring(tipo.indexOf(" ")+1),tipo.indexOf("l")-1);
            int lleve = Integer.parseInt(tipo.substring(tipo.lastIndexOf(" ")+1));

            int numpromo= (int)cantidad/lleve;
            if (numpromo==0) {

            }
            else {
            int cantidadpromo= numpromo*lleve;
            int cantnormal= (int)cantidad-cantidadpromo;
            double precionormal= cantnormal*producto.getPrecioUnitario();
            double preciopromo= numpromo*pague*producto.getPrecioUnitario();
            double preciofinal= precionormal+preciopromo;
            totalproducto=preciofinal;
            long idpromo = promocion.getId();
            actualizarPromocion(-numpromo,idpromo);}
      }

    }
```

Se obtiene el precio total de la compra del producto y se llama al método ps.adicionarVenta() para agregar la venta del producto a la tabla de ventas. Se agrega el subtotal del producto al contador que se inicio en el proceso y se agrega los datos de la venta del producto a la factura, posteriormente continua el ciclo hasta que se procesan todos los productos en el carrito.

```
Venta venta = ps.adicionarVenta(idcliente,idproducto,(int)cantidad,fecha,totalproducto,idsucursal,idventa);
preciototal+=totalproducto;
idventa= venta.getId();
factura += "\nId producto: "+idproducto+"\nNombre producto: "+
        producto.getNombre()+"\nCantidad: "+cantidad+"\nSubtotal producto: "+totalproducto;
```

Finalmente se limpia el carrito de los productos y se abandona.

```
ps.limpiarCarrito(idcarrito);
abandonarCarrito(correo);
return factura+"\nPrecio total: "+preciototal;
```

RF19

En la interfaz se va a pedir el correo del cliente que va a abandonar el carrito, con su correo se llama al método ps.darClientePorCorreo(correo), lo que nos va a devolver un objeto Cliente del cual sacaremos el id de este, con el id se obtendrá el carrito que tiene asignado, con el id del carrito se llamara al método ps.asignarCarrito(carrito.getId(), 0), se pasa el 0 por parámetro para

que el método haga que en la base de datos se refleje que el carrito no tiene ningún cliente asignado.

```
public long abandonarCarrito(String correo){
    Cliente cliente = ps.darClientePorCorreo(correo);
    long idcliente= cliente.getId();
    Carrito carrito =ps.darCarritoCliente(idcliente);
    ps.asignarCarrito(carrito.getId(), 0);
    return carrito.getId();

}
```

RF20

En la interfaz por input se pide el id del carrito del cual se van a devolver los productos, con este llamamos a la función darProductosCarrito() de la persistencia que nos va a devolver todos los productos que posee el carrito con su cantidad, se itera sobre cada producto y se obtiene el id y la cantidad que había en el carrito, con estos datos se llama a la función actualizarContieneEstante() de la persistencia que va a actualizar en la base de datos la cantidad que se va a encontrar en los estantes de un producto con estos devueltos. Finalmente después de terminar el ciclo se llama a la función limpiarCarrito de la persistencia que va a borrar de la base de datos la asociación del carrito con los producto que tenia.

```
public void limpiarCarrito(long idcarrito) {
    List<ContieneCarrito> carproductos= ps.darProductosCarrito(idcarrito);
    if(!carproductos.isEmpty()) {

        for(int i=0;i<carproductos.size();i++) {
            ContieneCarrito a =carproductos.get(i);
            long idproducto = a.getIdproducto();
            long cantidad= a.getCantidad();
            ps.actualizarContieneEstante(cantidad, idproducto);

        }
        ps.limpiarCarrito(idcarrito);

    }
}
```

RF21

Por la interfaz se pedirá por input el correo de la persona que intenta realizar el proceso, su clave y la fecha esperada para el pedido que se va a consolidar. Con el correo y la clave se darán 3 escenarios, en el primero la contraseña dada no corresponde con la contreaseña que se tiene en la base de datos del usuario, por lo que se acabara el proceso en ese momento. Otro caso es que el correo y contraseña si correspondan al usuario, pero el rol de este no sea gerente de sucursal por lo que no estará autorizado y se acabara ahí el proceso. En el ultimo caso el rol del usuario si corresponde a gerente de sucursal, se procederá a crear una lista vacia para meter los ids de los pedidos de la sucursal. Se obtendrá el id de la sucursal del usuario y se llamara al método

darPedidosPorSucursal() que devolverá una lista con las ordenes de pedido de esa sucursal. Se crear una lista vacia donde posteriormente se agregaran los id de los proveedores de los pedidos. Por medio de un ciclo, agregamos todos los proveedores a la lista obteniéndolos de la lista de ordenes de pedido. Posteriormente iteramos sobre la lista de ids de proveedores, para cada proveedor se van a obtener los pedidos específicos de ese proveedor para esa sucursal por medio del método darPedidosPorSucursalYProveedor(idsucursal, proveedores.get(i)). Vamos a iterar sobre estos pedidos obteniendo el precio de cada uno y agregándolo a un contando del precio final del pedido consolidado. Se procede a crear el pedido consolidado con sus datos con el método ps.consolidarPedido(), que además devolverá el id del pedido consolidado, con el cual procederemos a asginar el id para las ordenes de pedido por medio del método asignarPedidoConsolidado()

```
if(ps.darRolUsuarioPorId(usuario.getTipousuario()).getNombre().equals("Gerente de sucursal") ) {
    ArrayList<Long> idspedidos = new ArrayList<Long>();
    long idsucursal=usuario.getSucursal();
    List<Pedido>pdsucur=ps.darPedidosPorSucursal(idsucursal);
    ArrayList<Long> proveedores = new ArrayList<Long>();
    for(int i=0;i<pdsucur.size();i++) {
        long idpvd = pdsucur.get(i).getIdproveedor();
        if(!proveedores.contains(idpvd)) { proveedores.add(idpvd);}
    }
    for(int i=0;i<proveedores.size();i++) {
        double total=0;
        List<Pedido> peds = ps.darPedidosPorSucursalYProveedor(idsucursal, proveedores.get(i));
        System.out.print("cccc");
        for(int j=0;j<peds.size();j++) {

            total+=peds.get(j).getPreciosubtotal();
        }

        PedidoConsolidado pc =ps.consolidarPedido(proveedores.get(i),total,fechaesperada,idsucursal);

        ps.asignarPedidoConsolidado(idsucursal,proveedores.get(i),pc.getIdpedido());

        idspedidos.add(pc.getIdpedido());


    }
    log.info ("Se consolidaron los pedidos");
    return idspedidos;
```