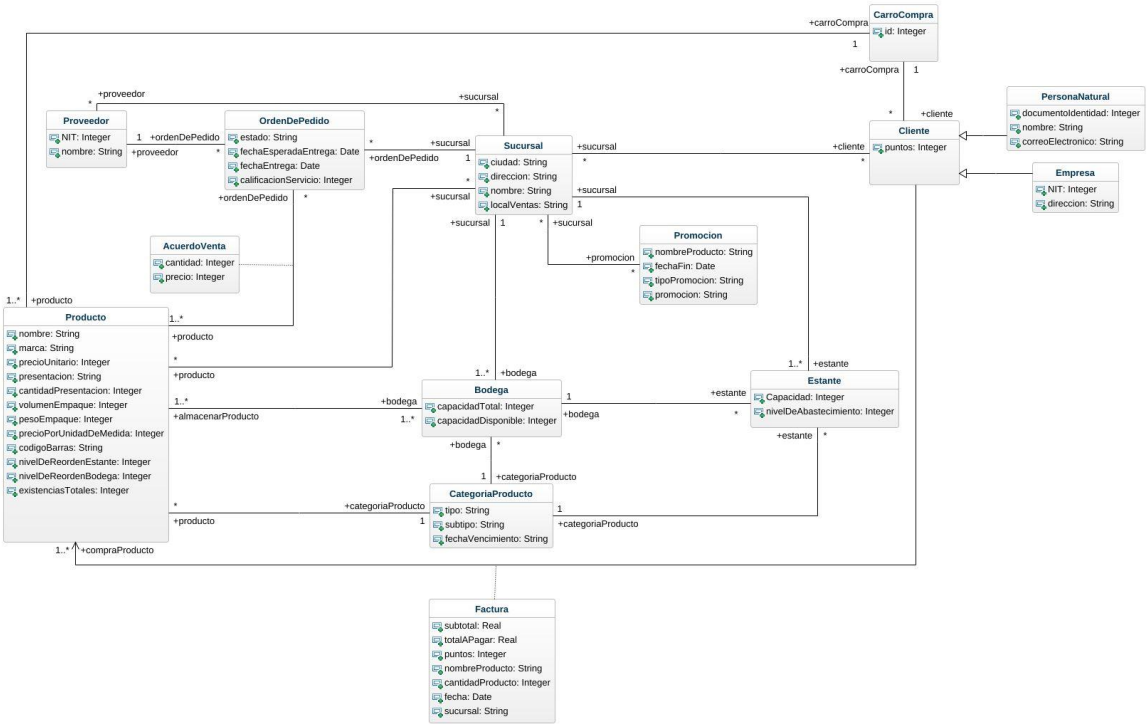
ITERACIÓN 3

Juan Camilo Neira

Luis Felipe Dussán Rueda

Análisis



Diseño de la aplicación

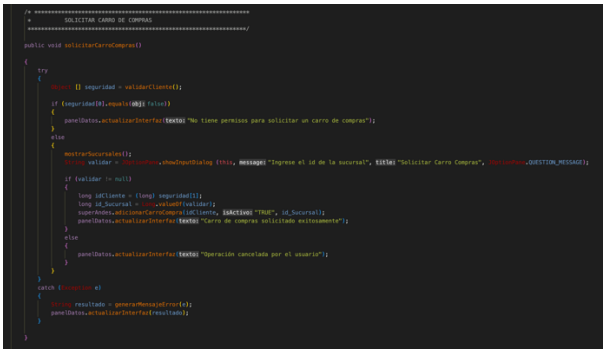
1. Impacto de los nuevos requerimientos y restricciones a nivel del modelo conceptual.

Para esta esta iteración introducimos la clase CarritoCompra, esto para satisfacer los nuevos requerimientos de forma más optima, con esta nueva clase el cliente puede añadir los productos al carrito, removerlos, pedir el carrito o pagarlo, con esta nueva metodología se vuelve más fácil para la aplicación y para los administradores devolver los productos que no fueron comprados. La introducción de esta nueva clase cambió las interacciones con otras clases, ya que la clase CarritoCompra interactúa de forma directa con otras clases como Bodega, Producto, Cliente, entre otras.

2. El modelo se encuentra en BCNF.

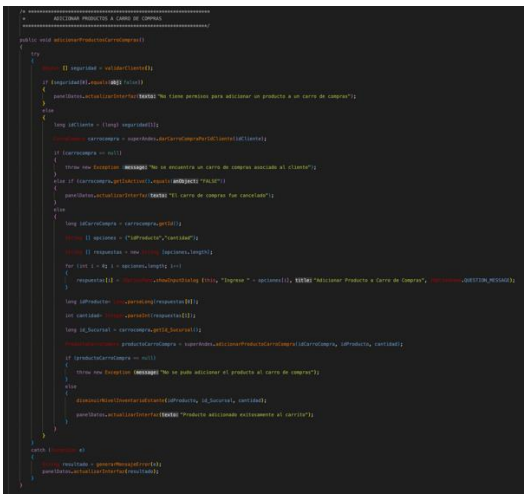
3. Lógica de los nuevos requerimientos a desarrollar.

   RFC1:



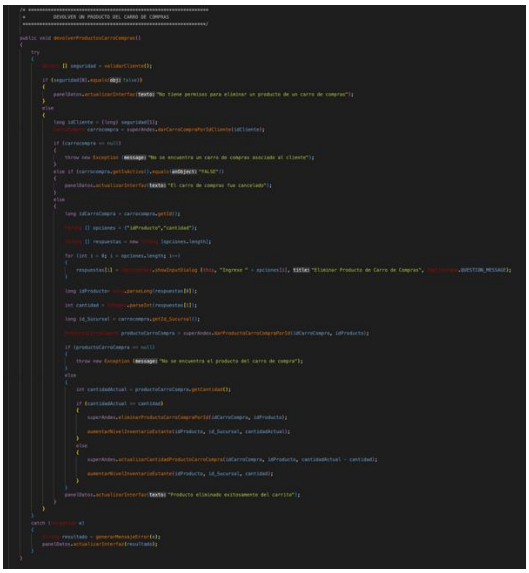
Resolvimos este requerimiento validando al cliente , solicitando al cliente el id de la sucursal y después la aplicación solicita el id del cliente y de la cliente, la aplicación se encarga de crear el carrito con esta información.

RFC2:



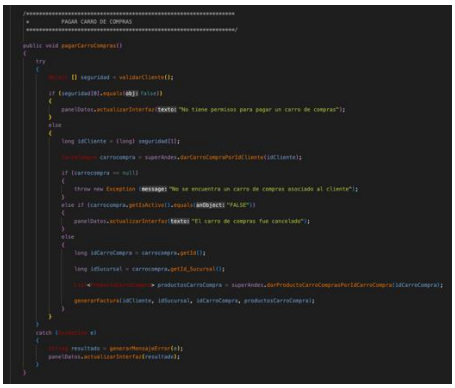
Para este requerimiento se valida al cliente, luego se busca si el cliente tiene asociado un carrito de compras, sino la aplicación responde con un error, después la aplicación pide el id del producto y la cantidad. Con esta información se añade el producto al carrito de compras, si se cancela la solicitud arroja un error. Después actualiza el inventario, reservando los productos que fueron añadidos.

RFC3:



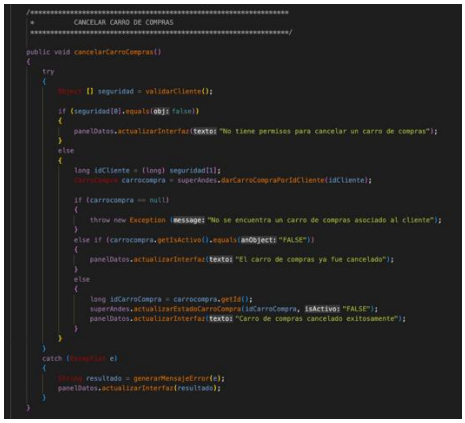
Para devolver un producto hay que validar el cliente, después validar que ese cliente tenga asociado un carrito de compras, la aplicación pide el producto y la cantidad que se quiere devolver, con esta información la aplicación elimina el producto del carrito y se encarga de devolverlo al inventario y de actualizar el estante.

RFC4:



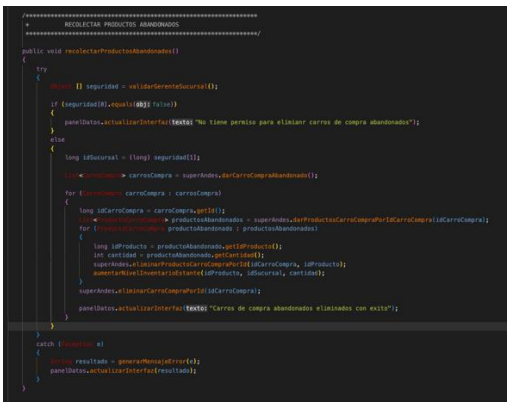
Para pagar el carro compra hay que validar primero al cliente, si tiene acceso prosigue, con el id cliente la aplicación busca el carrito de compras asociado, sino encuentra un carrito asociado manda error. Si tiene un carrito para pagar, la aplicación saca los productos del carrito y los mete en la factura, eliminándolos así del inventario.

RFC5:



Solucionamos este requerimiento, validando al cliente, si no tiene permiso arroja un error, después la aplicación con el id del cliente, busca el carrito de compra asociado, sino lo encuentra arroja otro error. Si pasa todos estos filtros, cambia el estado isActivo del carro a False, indicando que el carro fue abandonado.

RFC6:



En este requerimiento validamos al gerente de sucursal, si tiene credenciales la aplicación pide el id de la sucursal, después busca los carritos abandonados,
Por cada carrito busca los productos y su cantidad, los elimina del carro con su id y su cantidad, después actualiza el inventario y el estante, por último, la aplicación elimina el carrito.

RFC7:

```java
/* ************************************************************
 *          CONSOLIDAR PEDIDOS A LOS PROVEEDORES
 ************************************************************/

public void consolidarPedidosAProveedores()
{
    try
    {
        Object [] seguridad = validarGerenteSucursal();

        if (seguridad[0].equals(obj: false))
        {
            panelDatos.actualizarInterfaz(texto: "No tiene para recolectar productos abandonados");
        }
        else
        {


        }
    }
    catch (Exception e)
    {
        String resultado = generarMensajeError(e);
        panelDatos.actualizarInterfaz(resultado);
    }
}
```

RFC8:

Construcción de la aplicación.