**UNIVERSIDAD NACIONAL DE INGENIERIA**

FACULTAD DE CIENCIAS Y SISTEMAS



TESINA PARA OPTAR AL TÍTULO DE

INGENIERO DE SISTEMAS

Tema:

# PROTOTIPO DE SISTEMA DE FACTURACION Y CONTROL DE INVENTARIO, TIENDA DENMAR

PRESENTADO POR:

Andy Javier Quiroz Mejía 2001-10601

Horacio José Castillo Morales 98-11900-0

TUTOR

Ing. Patricia Lacayo Cruz

Managua, Nicaragua

Jueves 05 de Marzo del 2009

# Tienda DENMAR

## Contenido

## OBJETIVO GENERAL

- ✓ Elaborar un Prototipo de Sistema de Facturación y Control de Inventario basado en las actividades que se efectuan para la venta, compra y el control de los productos en la tienda DENMAR.

## OBJETIVOS ESPECÍFICOS

- ✓ Recopilar información concerniente al manejo y control de los artículos de la tienda, que nos permita realizar un estudio preliminar sobre el proceso del negocio que se sigue en la tienda DENMAR.

- ✓ Utilizar el Lenguaje de Modelación Unificado (UML) como base de compresión de las actividades operativas del negocio con el fin de facilitar el diseño del prototipo del sistema.

- ✓ Plantear una forma de almacenamiento de la información mas relevante que posee la tienda DENMAR.

- ✓ Elaborar una Interfaz Grafica de Usuario (GUI) para el Prototipo Sistema de Facturación y Control de Inventario en la tienda DENMAR.

- ✓ Proponer una distribución de red acorde a los requerimientos del Prototipo de Sistema de Facturación y Control de Inventario de la tienda DENMAR.

# JUSTIFICACION

Debido a la poca disponibilidad de tiempo que tienen las personas en la actualidad, nosotros hemos observado la necesidad de agilizar las transacciones que se efectúan en la tienda “DENMAR” lo cual se podría lograr mediante el prototipo de sistema de información que se esta proponiendo.

El motivo de la elaboración de este proyecto se basa en la necesidad del propietario del negocio con el obrero, de llevar eficiente y ágilmente el control de entradas y salidas de sus productos para proporcionarle información oportuna. Permitiéndole la toma de decisión con prontitud y la buena administración de su efectivo.

Podemos decir que dicho proyecto tiene un alto grado de importancia debido a que este se podrá plantear acorde a un proceso de negocio que llene los requerimientos para el buen funcionamiento del mismo en el corto y largo plazo, de tal forma que se permita adaptar a los cambios presentes y futuros siempre inminentes en las actividades de los negocios.

Todo lo anterior establecerá procedimientos de trabajo, que faciliten las referencias necesarias, para el conocimiento y apropiación de las distintas actividades desarrolladas en el negocio.

De manera general podemos decir, que con la terminación de este producto (Prototipo de Sistema de Facturación y Control de inventario) se logrará tener una base sólida a la hora de decidir automatizar los procesos más relevantes en la tienda.

Con la culminación de este proyecto se lograra lo siguiente:

- ✓ Analizar todas las debilidades existentes en la tienda en la actualidad.
- ✓ Plantear un método de almacenamiento de la información más relevante de la tienda.
- ✓ Proponer una solución de la interfaz de usuario del Prototipo de Sistema de Facturación y Control de Inventario.
- ✓ Establecer la posible distribución de red necesaria para implementar las aplicaciones y servicios que se utilizaran en la tienda.

## RESUMEN DEL TEMA

El presente proyecto esta basado en la elaboración de un prototipo de sistema de facturación y control de inventario en la tienda DENMAR, con el fin de facilitar el manejo de todas las actividades que se realiazan en la misma.

Para la elaboración de este proyecto se entrevisto al propietario de la tienda “DENMAR”, la cual nos brindo la información pertinente de cómo funciona el proceso desde la compra de los productos hasta la venta de los mismos en la tienda y a su vez se nos planteó algunas de las problemáticas que poseen dichos procesos. Todo lo anterior nos servirá como base para los requerimientos (tareas ha realizar) por parte del prototipo de sistema de información.

Se realizo un análisis de la forma de trabajo en la tienda (en lo que se refiere al proceso de facturación y control de existencia de los productos), así como los procedimientos que siguen para lograr todas sus tareas.

Con la elaboración de dicho análisis se logro visualizar todas las tareas que deben y se pudieron automatizar en la tienda.

Debido a que todos los proceso que se siguen en la actualidad en la tienda son empíricos y sin ningún registro escrito mas que el pago de los cargos impositivos a la instancia correspondiente, se utilizara la metodología RUP, para reflejar el proceso de facturación y control de inventario.

Se propuso un prototipo del sistema de facturación y control de inventario, con el propósito de ser analizado y ver si cumple con los requisitos del negocio.

# INTRODUCCION

Debido a las exigencias de modernización que están surgiendo en la actualidad, en este mundo de tecnología, en donde la necesidad esta en automatizar la mayoría de los procesos de negocios. Originando que las empresas se vuevan mas competitivas y ofrezcan mejores servicios, facilitando el trabajo en las tareas relevantes, permitiendo con esto, dedicar el tiempo a tareas más importantes.

La mayoría de las empresas hoy en día están interesadas en el tiempo que tarda en operar, las incertidumbres que se enfrentan al no tener acceso a su información de manera ágil han permitido que se busquen nuevos horizontes, para proyectarse como empresas capaces de brindar un servicio eficiente y eficaz, de igual manera facilitarle a sus clientes información precisa y ágil, sin tener que hacer las tareas empíricas que actualmente efectúan, ocasionándoles pérdidas financieras.

De esta necesidad ha surgido el desarrollo de tecnologías como son la elaboración de proyectos de software de calidad construidos con todos y cada uno de los requerimientos que la empresa necesita. La construcción de estos sistemas de información implica todo un proceso integral que contempla análisis exhaustivos del funcionamiento global de las empresas.

Es por esto que el análisis y diseño de sistemas de información, ejecutado por los analistas de sistemas, buscan examinar sistemáticamente las entradas o flujos de datos, el proceso de transformación de los datos, el almacenamiento de datos y la salida de la información dentro del contexto del negocio en particular. Además, es utilizado para analizar, diseñar e implementar mejoras en el funcionamiento de los negocios que pueden ser logrados por medio del uso de sistemas de información computarizados.

Los analistas de sistemas deben determinar la metodología que van ha utilizar para el desarrollo de los sistemas de información, en el presente trabajo utilizaremos RUP (Proceso Unificado Racional) como metodología de ingeniería de software. Esta metodología contempla las siguientes fases: análisis, diseño general, evaluación, diseño detallado, implementación y aceptación del sistema. A través de las fases anteriores se obtendrá como resultado la implementación de un prototipo de sistema de facturación y control de inventario.

Se elaboro un prototipo de sistema de información que administra de forma eficiente y eficaz el control de inventario y a su vez la facturación de la compra y venta de los productos en la tienda “DENMAR”. Con lo anterior mencionado se logro obtener una base de estudio que se tomo en cuenta a la hora de automatizar los procesos en la tienda.

La tienda “DENMAR” es una tienda que se dedica a la venta de una variedad de productos para personas en general, entre los cuales se encuentran; ropa y calzado. Todos los productos son vendidos al por menor. Actualmente en la tienda se esta considerando aumentar la variedad de productos, ya que esta piensa comenzar a ofertar todo lo referido a cosméticos y bisuterías.

Se realizo un estudio preliminar (Enfoque Sistémico) para encontrar la problemática que posee actualmente la tienda. Utilizamos la metodología RUP para realizar un análisis de cómo esta estructurado el prototipo de sistema de facturación y control de inventario propuesto en la tienda.

Se propuso la forma de almacenamiento de la información más relevante de la tienda mediante:

- ✓ El Diagrama de Clases: El cual contiene las tablas con los debidos atributos y métodos para trabajar con la información.

- ✓ Un Modelado de Datos: En donde se planteo el diseño físico propuesto de la base de datos, el cual esta montado con el gestor de base de datos Microsoft SQL SERVER 2000.

Se formuló el diseño del Modelo del Sistema de Información, el cual esta construido en base a los requerimientos obtenidos en la entrevista con el propietario de la tienda. El Prototipo de Sistema de Facturación y Control de Inventario para la Tienda DENMAR esta diseñado en NETBEANS 6.0.1 conectado con el gestor de base de datos Microsoft SQL SERVER 2000.

Para finalizar se presentó una estructura de red para las aplicaciones y recursos que se vayan a utilizar en la tienda DENMAR.

# ALCANCE DE PROYECTO

Los requisitos de información serán obtenidos directamente de los vendedores, en lo concerniente al procedimiento y las formalidades a la hora de efectuar una venta. Y del propietario de la tienda, en lo referido a las características de cada producto, el reabastecimiento (compra de productos) y los datos de los proveedores.

La manera que se recolectara toda la información necesaria será la siguiente:

- ✓ Cuestionario a los vendedores.
- ✓ Entrevistas con el propietario. (Ver Anexo I)

El Prototipo de Sistema de Facturación y Control de Inventario será capaz de lo siguiente:

- ✓ Llevar un control en lo referente a la venta y compra de productos.
- ✓ Emitir los recibos a los clientes y proveedores de la tienda.
- ✓ Realizar un control de inventario de los productos.
- ✓ Administrar las facturas de compra y venta de producto.
- ✓ Calcular los ingresos de venta por día, con lo cual se podrá hacer un arqueo con caja chica, para así emitir un informe de balance diario de la tienda, el cual se le entregara al propietario.
- ✓ Llevar un control de la información más relevante de los proveedores.
- ✓ Otras

El sistema se instalará en cinco computadoras, las cuales serán operadas por el propietario, el administrador, el facturador, el encargado de almacén y una para los vendedores.

El administrador será el encargado de realizar todas las tareas concernientes a la administración de un sistema (cuentas de usuario, niveles de acceso, respaldo de información, etc.).

Dentro de las limitaciones técnicas que enfrenta la tienda, se encuentran únicamente la necesidad de capacitar a los usuarios del prototipo del sistema de acuerdo a los niveles de acceso que posean.

# I MODELO DE REQUERIMIENTO

# 1 ENTORNO GENERAL DE LA EMPRESA

## 1.1 ANTECEDENTES

En la actualidad la existencia de establecimiento de ventas de enceres personales para minoristas se ha convertido en algo de mucha necesidad, tanto para los productores como para los clientes. Todo esto es debido a la calidad y forma de vida que se sigue en estos tiempos.

Debido a la poca disponibilidad que tiene las personas en la actualidad, al poco tiempo que poseen para las actividades personales cotidianas; todo esto es debido al alto grado de exigencias que hay actualmente, es que surgieron los establecimientos (tiendas) de ventas de artículos para minoristas. Debido a que en estos establecimientos se recibe una atención personalizada y en menor tiempo, ya que este es un factor importante en la actualidad.

El ahorro de tiempo y la disponibilidad de los productos, son los factores más importantes ha considerarse en este tipo de negocios (tiendas de ventas), es por esto la existencia de muchos de estos establecimientos en la actualidad y con esto surge la necesidad de disminuir el lapso de ejecución de la tareas, para lograr prestar un mejor y mas rápido servicio para los clientes, ya que estos lo que buscan obtener en este tipo de establecimiento es: un servicio de calidad, atención personalizada, disminución del tiempo de espera para adquirir las mercancías, etc.

La tienda DENMAR esta ubicada en el Barrio San José Oriental, de la Clínica Santa Maria 1C Arriba 1 1/2C al Sur. El nombre de la tienda proviene de una convinacion de los nombres de los dueños: DEN de Denis y MAR de Maritza. Dicha tienda surgió en el año 2006 como iniciativa propia de la propietaria Marlene Guido, quien después de renunciar a su trabajo, vio la necesidad de invertir el dinero obtenido de su liquidación y gozar así de independencia laboral. Inicialmente era la única persona trabajando en la misma, contratando en el siguiente año a una persona más que le ayudase en la venta y disposición de los productos. Más tarde contrato a personal para el apoyo en la administración y facturación de los artículos. Actualmente el negocio ha sufrido cambios en cantidad de personal y oferta productos, lo cual se detalla posteriormente (Situación Actual de la Empresa).

## 1.2 OBJETIVOS DE LA ORGANIZACIÓN

Debido al tipo de negocio o giro de la tienda, sus objetivos como organización son los siguientes:

- ✓ Prestar un servicio personalizado de calidad.
- ✓ Mostrar agilidad en sus procesos o tareas a realizar
- ✓ Proveer una variedad de mercancías a sus clientes
- ✓ Dar una mejor disponibilidad de los productos y proporcionar los mismos en un menor tiempo.

## 1.3 ORGANIGRAMA GENERAL ACTUAL

No posee un sistema organizacional, ya que los empleados no tienen las tareas bien establecidas, es decir no llevan un orden a la hora de realizar las tareas.

## FUNCIONES GENERALES DE LOS EMPLEADOS ACTUALMENTE

**Propietario:**

- ✓ Analizar las negociaciones de los diferentes productos para su comercialización.
- ✓ Hacer pedido y Pagos a los proveedores.
- ✓ Controla el comportamiento de las ventas para obtener los mínimos y máximos existencias de ventas.
- ✓ Reuniones con proveedores
- ✓ Discutir con diferentes niveles de financiamiento (crédito) para realizarse o crecer como empresa, atender a los clientes (en Ocasiones).
- ✓ Informar al vendedor sobre los precios de cada producto.
- ✓ Calcular los precios de venta de los productos.

**Facturador:**

- ✓ Realizar factura comercial a los clientes.
- ✓ Informar al propietario de los productos agotados, para que este haga el pedido a los proveedores.
- ✓ Se hace cargo de registrar los ingresos y egresos.
- ✓ Realiza control de inventario con ayuda del vendedor.
- ✓ Realizar informe de ventas diarias para la dueña
- ✓ Realiza informe de productos faltantes para ser pedidos a los proveedores.
- ✓ Realiza balance diario de Caja y libro diario.

**Vendedor:**

- ✓ Definir cuales son las necesidades de los clientes.
- ✓ Atender a los clientes, empacar los productos a los clientes
- ✓ Acomodar productos en estantes
- ✓ Ayudar en el conteo de mercancías (Control de Inventario).
- ✓ Es encargado del Mantenimiento y limpieza de la tienda, etiquetar los productos.

# 2 DESCRIPCION DEL PROBLEMA

## 2.1 SITUACIÓN ACTUAL DE LA EMPRESA

De forma general los aspectos más importantes que se lograron ver con la entrevista: es que la tienda no posee una estructura organizacional y todos los procesos los realizan manualmente. Actualmente en la tienda, el gerente propietario es el encargado de las tareas más relevantes, las cuales son los pedidos y pagos a los proveedores, atender las diferentes negociaciones con los proveedores, en ocasiones el mismo tiene que atender a la clientela.

La tienda cuenta con tres empleados, los cuales son:

- ✓ **Dos vendedores:** que se encargan de atender a los clientes, coloca los precios a los productos.
- ✓ **Un encargado de facturación:** registrar las ventas en el libro diario, lleva el control de inventario con ayuda de los vendedores, realiza los reportes diarios de las ventas, revisa los pedidos a los proveedores, acomoda los productos en los estantes y realiza informe de productos agotados.

Para el control de inventario se hacen conteos manuales en fechas no definidas, para las compras se lleva un registro diario el cual es realizado por el facturador.

A la hora de realizar una venta el facturador es el encargado de registrar en el libro diario dicha transacción, en el cual se anota la fecha de la venta, la cantidad vendida y el monto total de la venta. En el caso de que el cliente solicite descuento, este solo es autorizado por el propietario.

Los productos en general que ofrece la tienda son los siguientes: ropa para dama y caballero, cosméticos para damas, cortes de tela, etc. Los cuales se ofrecen por docena o unidad y en el caso de los corte por yarda, metro, centímetros. Con lo que se refiere a la adquisición de mercancías (compra a proveedores). Los productos entrantes se registran en un cuaderno, en el cual se anotan la fecha y el proveedor a quien le compren los productos, luego se listan los productos comprados, las cantidades y el monto total por cada producto.

El facturador es el encargado de calcular el precio unitario y el precio de venta, luego el pasa el informe al propietario, el cual revisa el informe, si todo le parece bien procede a pagarle al proveedor. Para calcular los precios de venta el facturador aplica el 15% de I.V.A.

Todas las compras se hacen al por mayor. No se hacen restricciones sobre compras de los productos.

Al final del día se suman todas las ventas y así se calcula la venta total del día, esto es comparado con el dinero contenido en la caja.

En lo que se refiere al prototipo de sistema que se va implementar, la opinión del propietario es, que sí, le interesaría para que así se agilicen la facturación y control de inventario, ya que con esto se presta un mejor servicio y serán más competitivos con respectos a otros competidores más cercanos.

Con respecto a la experiencia que existe de parte de los empleados en el uso de software, obtuvimos lo siguiente:

- ✓ El Propietario: Estudio de un curso de computación, sobre los programas que se emplean comúnmente para procesar texto y manipular información relacionada con cálculos matemáticos: Microsoft Office. Conoce como acceder a Internet y manejo de correo electrónico.

Pero su objetivo es contratar o adiestrar a uno de sus trabajadores para que se encargue del manejo del sistema de información. A la ahora que el sistema este montado las únicas personas que los manipularan serán: La persona que será contratada o adiestrada y el propietario.

## 2.2 SITUACIÓN PROBLEMICA DE LA EMPRESA

**(Un enfoque Sistémico)**

**Descripción del problema**

La principal problemática que enfrenta la tidenda DENMAR es la siguiente:

- ✓ La insatisfacción de los clientes debido a la lentitud con que se prestan los servicios, lo cual conyeva a obstaculizar el desarrollo de la entidad y a su vez disminuye la clientela que visita la tienda, lo cual se convierte en perdidas cuantiosas para la misma.

El problema anteriormente mencionado tiene su origen en varios factores que se mencionan a continuación: La tienda “DENMAR” en la actualidad es un negocio que no cuenta con una estructura organizacional bien definida, ni un sistema automatizado que lleve el control de las entradas y salidas de inventario y que realice las facturas a los clientes en el momento de efectuar las compras de los productos.

Este problema se dá ya que existe poco interés y falta de visión por parte de la propietaria de volverse una empresa más competitiva con respecto a sus competidores, lo cual ha provocado que en la tienda las tareas se vuelvan muy tediosas y a su vez que se realicen en mayor cantidad de tiempo.

De forma específica los dos principales puntos a considerar a la hora de analizar la problemática de la tienda serian:

- ✓ La falta de una estructura organizacional bien definida, lo cual provoca un menor nivel en el desempeño por parte de los empleados, esto se da por que no poseen sus funciones de trabajo bien definidas.

- ✓ La falta de un sistema automatizado, ya que desde su inauguración todo se realiza manualmente, lo que provoca que el proceso se vuelva lento ya que la tienda se esta expandiendo en clientela y cantidad de productos que ofrecen a sus clientes.

Los efectos causados por lo anteriormente mencionado son los siguientes:

- ✓ Debido a que todo se hace manualmente se tiene un gran porcentaje a cometer errores.
- ✓ El proceso de facturación de la compra y venta de los productos es muy tardado.
- ✓ El control de inventario se hace muy complejo (Pesado).
- ✓ Fácil perdida de información por que todo solo esta plasmado en papeles.
- ✓ Errores en los cálculos de facturas en casos de facturar y controlar inventario.
- ✓ La tienda posee desventaja competitiva con respecto a las tiendas Cercanas que poseen un Sistema de información Automatizado.
- ✓ Mal desempeño en las funciones de los empleados.
- ✓ Perdidas cuantiosas para la tienda debido a la insatisfacción en los clientes.

**SOLUCIONES PLANTEADAS.**

1. Invertir en el S.I ya sea haciendo un préstamo o asignar recursos de las ganancias de las ventas.
2. Reunirse con el propietario para hacerle ver los beneficios de un S.I.
3. Reorganizar la empresa de tal forma que cada empleado tenga una función específica y así prestar un servicio más eficiente.
4. Implementar un S.I con lo que se disminuirá el tiempo de cálculo y los errores para el manejo de inventario, la facturación se agilizara y por ende se reducirá el tiempo de espera de los clientes.
5. Redistribuir las funciones de los empleados de forma mas especifica.
6. Contratar más personal.

## 2.3 DEFINICION DE REQUERIMIENTOS DEL SISTEMA

### Introducción a SQL Server 2000

SQL Server 2000 es un sistema de gestión de bases de datos relacionales (SGDBR o RDBMS: Relational Database Management System) diseñado para trabajar con grandes cantidades de información y la capacidad de cumplir con los requerimientos de proceso de información para aplicaciones comerciales y sitios Web. SQL Server 2000 ofrece el soporte de información para las tradicionales aplicaciones Cliente/Servidor, las cuales están conformadas por una interfaz a través de la cual los clientes acceden a los datos por medio de una LAN.

La hoy emergente plataforma NET exige un gran porcentaje de distribución de recursos, desconexión a los servidores de datos y un entorno descentralizado, para ello sus clientes deben ser livianos, tales como los navegadores de Internet los cuales accederán a los datos por medio de servicios como el Internet Information Services(IIS).

SQL Server 2000 está diseñado para trabajar con dos tipos de bases de datos :

- ✓ **OLTP (OnLine Transaction Processing):** Son bases de datos caracterizadas por mantener una gran cantidad de usuarios conectados concurrentemente realizando ingreso y/o modificación de datos. Por ejemplo : entrada de pedidos en línea, inventario, contabilidad o facturación.
- ✓ **OLAP (OnLine Analytical Processing):** Son bases de datos que almacenan grandes cantidades de datos que sirven para la toma de decisiones, como por ejemplo las aplicaciones de análisis de ventas.

SQL Server puede ejecutarse sobre redes basadas en Windows Server así como sistema de base de datos de escritorio en máquinas Windows NT Workstation, Windows Millenium y Windows 98.

Los entornos Cliente/Servidor, están implementados de tal forma que la información se guarde de forma centralizada en un computador central (servidor), siendo el servidor responsable del mantenimiento de la relación entre los datos, asegurarse del correcto almacenamiento de los datos, establecer restricciones que controlen la integridad de datos, etc.

Del lado cliente, este corre típicamente en distintas computadoras las cuales acceden al servidor a través de una aplicación, para realizar la solicitud de datos los clientes emplean el Structured Query Language (SQL), este lenguaje tiene un conjunto de comandos que permiten especificar la información que se desea recuperar o modificar.

Existen muchas formas de organizar la información pero una de las formas más efectivas de hacerlo está representada por las bases de datos relacionales, las cuales están basadas en la aplicación de la teoría matemática de los conjuntos al problema de la organización de los datos. En una base de datos relacional, los datos están organizados en tablas (llamadas relaciones en la teoría relacional).

Una tabla representa una clase de objeto que tiene importancia para una organización. Por ejemplo, se puede tener una base de datos con una tabla para empleados, otra para clientes y otra para productos del almacén. Las tablas están compuestas de columnas y filas (atributos y tuplas en la teoría relacional).

La tabla Empleados tendría columnas para el nombre, el apellido, código del empleado, departamento, categoría laboral y cargo. Cada fila representa una instancia del objeto representado por la tabla. Por ejemplo, una fila de la tabla Empleados representa el empleado cuyo Id. de empleado es 12345.

Al organizar los datos en tablas, se pueden encontrar varias formas de definirlas. La teoría de las bases de datos relacionales define un proceso, la normalización, que asegura que el conjunto de tablas definido organizará los datos de manera eficaz.

SQL Server incluye un conjunto de herramientas que facilitan la instalación y administración del servidor así como un conjunto de herramientas que facilitan el diseño e implementación de base de datos, entre ellos podemos mencionar:

- ✓ SQL Server 2000 Database Engine, diseñado para almacenar detalladamente los registros de las operaciones transaccionales (OLTP), este motor es responsable de mantener la seguridad de los datos, proveer un adecuado nivel de tolerancia a fallos, optimizar las consultas, emplear adecuadamente los bloqueos de recursos para optimizar la concurrencia, etc.

- ✓ SQL Server 2000 Analysis Services, provee herramientas para consultar información almacenada en data warehouses y data marts, como por ejemplo cuando se desea obtener información totalizada acerca de los niveles de ventas mensuales por regiones de ventas, etc.

Soporte para aplicaciones, SQL Server brinda a las aplicaciones clientes la posibilidad de acceder a los datos a través de un lenguaje denominado Transact-SQL, asimismo es importante mencionar que ahora existe un soporte para devolver la información en formato XML.

Entre los componentes adicionales de SQL Server 2000, podemos mencionar:

Data Transformation Services, permite recuperar información de un origen de datos, realizar transformaciones sencillas o complejas (como totalización de datos) y almacenarlos en otro origen de datos, como una base de datos SQL o un cubo multidimensional.

Replicación, se puede distribuir la información a través de un mecanismo de replicación con la finalidad de optimizar el rendimiento o de mantener autonomía, mientras una copia de la información almacenada en diferentes computadoras mantengan sincronización.

English Query, provee de un sistema que permite a los usuarios plantear una pregunta en lenguaje natural en lugar de emplear un formato Transact-SQL. Por ejemplo: “List all customers”, “How many  blue dress were sold in 2001?”, etc.

Meta Data Services, son un conjunto de servicios que permiten almacenar información acerca de las bases de datos y aplicaciones clientes que las emplean, esta información es aprovechada cuando se requiere intercambiar con otras aplicaciones. Los Meta Data Services proveen tres estándares: Meta Data Coalition Open Information Model (MDC OIM), Interfaces COM y XML Encoding.

**PROGRAMACION EN JAVA**

Java es un nuevo lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. Sun describe al lenguaje Java de la siguiente manera: Simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portable, de alto rendimiento, multitarea y dinámico.

Sun admite totalmente que lo dicho anteriormente es una cadena de halagos por parte suya, pero el hecho es que todo ello describe al lenguaje Java.

Java permite hacer cosas excitantes con las paginas Web que antes no eran posibles. De manera que en este momento la gran interactividad que proporciona Java marca la diferencia en las paginas Web. Imagina un Web donde puedes jugar a un juego, como el fútbol, tú y otras personas que están en lugares remotos forman parte de un equipo y otras mas del contrario, verías a los jugadores animados en la pantalla obedeciendo las instrucciones de las personas que están jugando al juego desde sitios remotos.

Además las puntuaciones quedarían registradas. O un Web con una aplicación en donde el usuario pueda hacer transacciones y estas se actualicen en tiempo real. O un sitio que ofrezca pequeñas aplicaciones como hojas de calculo o calculadoras para uso de los visitantes. O uno que muestre figuras 3D, tales como moléculas o dinosaurios que pueden ser rotados con un click del ratón.

Java también aumenta el contenido multimedia de un sitio, ofreciendo animaciones fluidas, gráficos mejorados, sonido y vídeo, fuera de lo necesario para enganchar aplicaciones de ayuda dentro de sus navegadores Web.

Desde J2SE 1.4, la evolución del lenguaje ha sido regulada por el JCP (Java Community Process), que usa Java Specification Requests (JSRs) para proponer y especificar cambios en la plataforma Java. El lenguaje en sí mismo está especificado en la Java Language Specification (JLS), o Especificación del Lenguaje Java. Los cambios en los JLS son gestionados en JSR 901.

- ✓ JDK 1.0 (23 de enero de 1996) — Primer lanzamiento.

- ✓ JDK 1.1 ( de 1997) — Principales adiciones incluidas:
  - Una reestructuración intensiva del modelo de eventos AWT (Abstract Windowing Toolkit)
  - Clases internas (inner classes)
  - JavaBeans
  - JDBC (Java Database Connectivity), para la integración de bases de datos
  - RMI (Remote Method Invocation)

- ✓ J2SE 1.2 (8 de diciembre de 1998) — Nombre clave Playground. Esta y las siguientes versiones fueron recogidas bajo la denominación Java 2 y el nombre "J2SE" (Java 2 Platform, Standard Edition), reemplazó a JDK para distinguir la plataforma base de J2EE (Java 2 Platform, Enterprise Edition) y J2ME (Java 2 Platform, Micro Edition). Otras mejoras añadidas incluían:
  - Reflexión en la programación
  - La API gráfica ( Swing) fue integrada en las clases básicas
  - La máquina virtual (JVM) de Sun fue equipada con un compilador JIT (Just in Time) por primera vez
  - Java Plug-in
  - Java IDL, una implementación de IDL (Interfaz para Descripción de Lenguaje) para la interoperabilidad con CORBA
  - Colecciones (Collections)

- ✓ J2SE 1.3 (8 de mayo de 2000) — Nombre clave Kestrel. Los cambios más notables fueron:
  - La inclusión de la máquina virtual de HotSpot JVM (la JVM de HotSpot fue lanzada inicialmente en abril de 1999, para la JVM de J2SE 1.2)
  - RMI fue cambiado para que se basara en CORBA
  - JavaSound

- Se incluyó el Java Naming and Directory Interface (JNDI) en el paquete de librerías principales (anteriormente disponible como una extensión)
- Java Platform Debugger Architecture (JPDA)

✓ J2SE 1.4 (6 de febrero de 2002) — Nombre Clave Merlin. Este fue el primer lanzamiento de la plataforma Java desarrollado bajo el Proceso de la Comunidad Java como JSR 59. Los cambios más notables fueron:

- Expresiones regulares modeladas al estilo de las expresiones regulares Perl
- Encadenación de excepciones Permite a una excepción encapsular la excepción de bajo nivel original.
- non-blocking NIO (New Input/Output) (Especificado en JSR 51.)
- Logging API (Specified in JSR 47.)
- API I/O para la lectura y escritura de imágenes en formatos como JPEG o PNG
- Parser XML integrado y procesador XSLT (JAXP) (Especificado en JSR 5 y JSR63.)
- Seguridad integrada y extensiones criptográficas (JCE, JSSE, JAAS)
- Java Web Start incluido (El primer lanzamiento ocurrió en Marzo de 2001 para J2SE 1.3) (Especificado en JSR 56.)

✓ J2SE 5.0 (30 de septiembre de 2004) — Nombre clave: Tiger. (Originalmente numerado 1.5, esta notación aún es usada internamente.) Desarrollado bajo JSR 176, Tiger añadió un número significativo de nuevas características

- Plantillas (genéricos) — provee conversion de tipos (type safety) en tiempo de compilación para colecciones y elimina la necesidad de la mayoría de conversion de tipos (type casting). (Especificado por JSR 14.)

- Metadatos — también llamados anotaciones, permite a estructuras del lenguaje como las clases o los métodos, ser etiquetados con datos adicionales, que puedan ser procesados posteriormente por utilidades de proceso de metadatos. (Especificado por JSR 175.)
- Autoboxing/unboxing — Conversiones automáticas entre tipos primitivos (Como los int) y clases de envoltura primitivas (Como Integer). (Especificado por JSR 201.)
- Enumeraciones — la palabra reservada enum crea una typesafe, lista ordenada de valores (como Dia.LUNES, Dia.MARTES, etc.). Anteriormente, esto solo podía ser llevado a cabo por constantes enteras o clases construidas manualmente (enum pattern). (Especificado por JSR 201.).
- Varargs (número de argumentos variable) — El último parámetro de un método puede ser declarado con el nombre del tipo seguido por tres puntos (e.g. void drawtext(String... lines)). En la llamada al método, puede usarse cualquier número de parámetros de ese tipo, que serán almacenados en un array para pasarlos al método.
- Bucle for mejorado — La sintaxis para el bucle for se ha extendido con una sintaxis especial para iterar sobre cada miembro de un array o sobre cualquier clase que implemente Iterable, como la clase estándar Collection.

✓ Java SE 6 (11 de diciembre de 2006) — Nombre clave Mustang. Estuvo en desarrollo bajo la JSR 270. En esta versión, Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión. Está disponible en http://java.sun.com/javase/6/. Los cambios más importantes introducidos en esta versión son:

- Incluye un nuevo marco de trabajo y APIs que hacen posible la combinación de Java con lenguajes dinámicos como PHP, Python, Ruby y JavaScript.

- Incluye el motor Rhino, de Mozilla, una implementación de Javascript en Java.
- Incluye un cliente completo de Servicios Web y soporta las últimas especificaciones para
- Servicios Web, como JAX-WS 2.0, JAXB 2.0, STAX y JAXP.
- Mejoras en la interfaz gráfica y en el rendimiento.

✓ Java SE 7 — Nombre clave Dolphin. En el año 2006 aún se encontraba en las primeras etapas de planificación. Se espera que su desarrollo dé comienzo en la primavera de 2006, y se estima su lanzamiento para 2008.

- Soporte para XML dentro del propio lenguaje.
- Un nuevo concepto de superpaquete.
- Soporte para closures.
- Introducción de anotaciones estándar para detectar fallos en el software.

✓ No oficiales:

- NIO2
- Java Module System.
- Java Kernel.
- Nueva API para el manejo de Dias y Fechas, la cual reemplazara las antiguas clases Date y Calendar. Posibilidad de operar con clases BigDecimal usando operandos.

## JAVA GUI

Los entornos graficos de usuario (GUI por sus siglas en ingles), son un tipo de interfaz de usuario que permiten interactuar facil y directamente con un dispositivo generalmente un ordenador. Se basa en elementos visuales amigables, simples y de rapido entendimiento para presentar y obtener informacion.

Las GUI hoy en dia estan en todas partes en equipos moviles (PDA, celulares, PMC, etc), en ordenadores (Windows, Linux, Mac), en cajeros automaticos, dipositivos especiales como (RMI, CAT scan), y cada dia ganan aun mas popularidad, la competencia por tener un mayor acercamiento y sastifacer las necesidades visuales de los cada vez mas exigentes usuarios continua. Durante esta carrera de necesidades los lenguajes de programacion han evolucionado, facilitando el manejo, creacion y reutilizacion de interfaces visuales.

Las herramientas mas avanzadas actualmente permiten la facil creacion de estas interfaces permitiendo al programador enfocarse en detalles de posiciones y de presentacion de informacion, este tipo de herramienta son denominadas RAD (Rapid Application Development) que no solo contienen la capacidad de crear GUI en tiempos record, si no tambien el generar codigos genericos, deteccion de errores en tiempo de compilacion mas rapidamente, facilitar la navegacion entre codificaciones.

Java ha podido facilitar el desarrollo de GUI con su rapida evolucion, desde la aparicion de sus primeros frameworks de desarrollo AWT hasta lo mas ultimo SWING y SWT.

**Programación en GUIs**

La programacion de interfaces visuales en los ultimos años ha mejorado considerablemente antes de continuar, es necesario conocer las herramientas necesarias:

- ✓ J2SE 1.5 o mayor, este es el mnemonico para Java Standard Edition de segunda generacion, este contiene el Java Developtment Kit para desarrollo osea para programar, y el Java Virtual Machine para ejecutar. (Recuerde que son tres corrientes en el mundo JAVA, J2SE, J2ME y J2EE, no mal interprete ninguna).
- ✓ Un editor de texto cualquiera, siempre y cuando se sepa compilar desde consola, shell o terminal. Es posible utilizar alternativas como netbeans, Eclipse, JBuilder o IntelliJ, los ultimos dos son comerciales.
- ✓ Conocimiento basicos de programacion Java, deseos de aprender y un ordenador.

Como deciamos el continuo desarrollo de herramientas como Netbean el poderoso IDE Java permiten ahora crear interfaces visuales extremadamente rapida, que agilizan algunas tareas. Netbeans 5 o mayor, un IDE oficial de Sun Microsystem, y cuenta con una poderosa herramienta RAD (Rapid Application Development, en castellano son herramientas de arrastre de componentes seleccionados de una paleta), que igualmente es criticada por su excesiva codificacion y su extenso uso de su formato basado en XML, o Form para su correcta aplicación

## A. REQUERIMIENTOS DE HARDWARE

Debido a la complejidad de los software que se van a implementar para la elaboración del Prototipo de Sistema de Facturacion y Control de Inventario en la Tienda DENMAR las especificaciones de los equipos donde se va ha montar el prototipo de sistema son las siguientes:

**Tomando en cuenta SQL SERVER 2000**

Antes de instalar SQL Server 2000 es necesario conocer cuales son los requisitos mínimos para instalar este producto, el siguiente cuadro muestra los requerimientos para instalar SQL Server de acuerdo a la edición que Ud. emplee:

| Recurso | Requerimiento |
|---|---|
| Computador | Intel o compatible |
| Procesador | Pentium 166 |
| Monitor | 800*600 |
| Dispositivo puntero | Mouse |
| Tarjeta de red | Opcional (requerido para acceso a los recursos de la red) |
| CD-ROM | Requerido para la instalación |

Para determinar correctamente el requerimiento de memoria, emplear la siguiente tabla:

| | **Enterprise** | **Estándar** | **Evaluation** | **Developer** | **Personal y Desktop Engine** |
|---|---|---|---|---|---|
| Alguna edición de Windows 2000 Server | 256 MB (128 MB soportado) | 256 MB (128 MB soportado) | 256 MB (128 MB soportado) | 256 MB (128 MB soportado) | 256 MB (128 MB soportado) |
| Alguna edición de Windows NT 4.0 Server con SP5 o posterior | 128 MB (64 MB soportado) | 64 MB | 128 MB recomendado (64 MB soportado) | 64 MB | 32 MB |
| Windows 2000 Professional | N/A | N/A | 128 MB recomendado (64 MB soportado) | 64 MB | 64 MB |
| Windows NT 4.0 Workstation, con SP5 o posterior | N/A | N/A | 128 MB recomendado (64 MB soportado) | 64 MB | 32 MB |
| Windows ME | N/A | N/A | N/A | N/A | 32 MB |
| Windows 98 | N/A | N/A | N/A | N/A | 32 MB |

## B. REQUERIMIENTOS DE SOFTWARE

**Tomando en cuenta SQL SERVER 2000**

Se debe tener en cuenta que para instalar SQL Server 2000 se requiere de Internet Explorer 5.0 o posterior, si desea instalar SQL Server 2000 sobre Windows NT en cualquiera de sus ediciones debe instalar previamente el Service Pack 5.0 o posterior. Asimismo tenga en cuenta la siguiente tabla, para poder determinar el espacio en disco requerido para su instalación:

| Opción de Instalación seleccionada | Espacio en disco requerido |
|---|---|
| Server y client tools | 95-270 MB dependiendo de las opciones seleccionadas |
| Instalación Typical | 250 MB (178 MB para el sistema, más 72 MB para programas y archivos de datos) |
| Instalación mínima | 110 MB (73 MB para el sistema, más 37 MB para programas y archivos de datos) |
| Herramientas administrativas | 113 MB (sistema solamente) |
| BoAceptars Online | 30 MB (sistema solamente) |
| Analysis Services | 47 MB mínimo 120 MB typical |
| English Query | 80 MB |
| Sólo Desktop Engine | 44 MB |

**Tomando en cuenta NET BEANS 6.0**

Lo primero que se debe saber sobre la instalación y los requerimientos sobre el IDE (Integrated Development Environmen) o Entorno de Desarrollo Integrado por sus siglas en ingles de este software. Para poder instalarlo se debe tener instalado previamente:

- JVM (Java Virtual Machine) o la Maquina Virtual de Java que es la traduce el código binario al código de maquina.

- JDK (Java Development Kit) o Kit (Set) para el Desarrollo de Java que contiene todas las librerías y herramientas para correr aplicaciones java como compiladores y interfaces para la detección de errores que son necesarios para el desarrollo de aplicaciones y applets. Usualmente este viene con las siglas SE (Standard Edition) que significa que es una edición estándar. Existe de esta manera y ED (Enterprise Edition) o para empresas.
- Tener instalada la versión de Net Beans 6.0.
- Instalar sql2ksp3, el cual es un parche para SQL 2000 SERVER en ingles para poder utilizarlo con NetBeans 6.0.

## C. REQUERIMIENTOS USUARIOS

**Factrurador:** Tener acceso a todas las pantallas del Prototipo de Facturación y Control de Inventario, a las impresoras.

**Administrador:** Tener acceso a SQL SERVER 2000, a las impresoras, a todas las pantallas del prototipo del sistema y al código fuente del mismo.

**Bodega:** Tener acceso a todas las pantallas referentes con el control de los productos.

**Propietario:** Tener acceso solo para hacer consultas en el prototipo del sistema, a las impresoras.

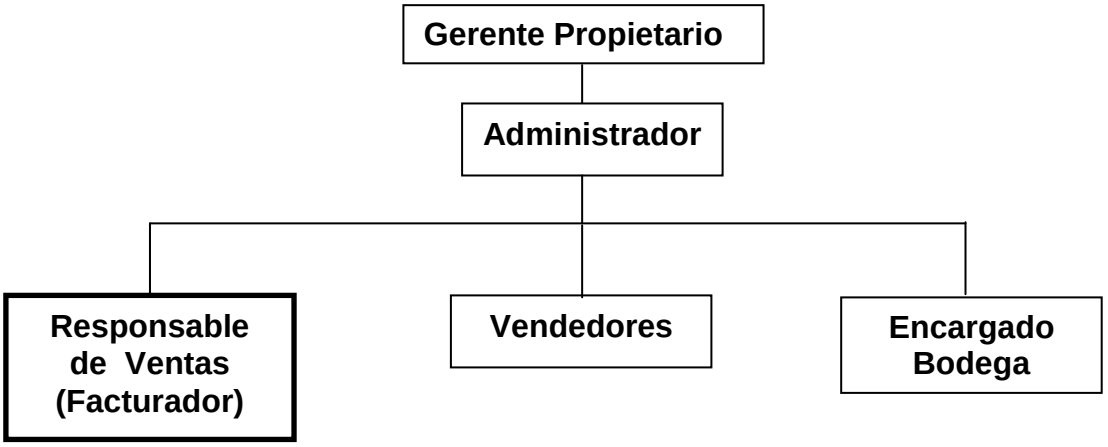
**Vendedor:** Tener acceso solo para hacer consultas en el prototipo del sistema.

## 2.4 ESTRUCTURA PROPUESTA DE LA EMPRESA

Se recomienda contratar más personal en la tienda para que se distribuyan equitativamente las tareas en la misma. Las personas que se recomienda contratar son las siguientes:

- ✓ **Un Administrador:** Es el que va estar encargado de la tienda en ausencia del propietario, será el encargado de administrar el sistema y la red que se va ha proponer, llevar el control de todos los tramites contables que se efectúen en la tienda.

- ✓ **Un encargado de Bodega:** Sus únicas funciones son: almacenar los productos en la bodega, catalogar los productos (Bien y Mal estado), llevar un control de los productos en bodega



# 3 DESCRIPCION DEL SISTEMA DE NEGOCIOS

La tienda “DENMAR” es una empresa que nace de las necesidades de los clientes de un cierto tipo de productos en un menor tiempo. Actualmente la tienda no cuenta con un sistema automatizado sino con un sistema manual donde registra sus compras y ventas en un libro diario y en donde tiene la lista de los proveedores y clientes. Para su control de inventario el encargado de la facturación con ayuda de los vendedores hacen conteos manuales esporádicos.

**Para realizar los pedidos a los proveedores:** la dueña le solicita al encargado de facturación que le de la lista de los productos que están por agotarse o se encuentran agotados. Luego la dueña hace una llamada a los proveedores para realizar el pedido. Por parte del proveedor: Si posee los productos reflejados en el pedido, este entrega los productos a la tienda en caso contrario llama al propietario informando que no tiene los productos que esta solicitando.

Cuando el pedido ya esta en la tienda, el Facturador es el encargado de recibir el pedido y luego revisarlo detalladamente para observar si no hay ninguna anomalía. Pueden haber dos tipos de anomalías: los productos están dañados, los productos no coinciden con el pedido hecho por el propietario.

Si el pedido coincide con lo pedido por proveedor el Facturador informa al propietario para que este realice el pago al proveedor, en el caso de anomalías el Facturador informa a la propietaria la cual realiza los arreglos pertinentes con el proveedor y luego procede a pagarle al mismo.

**Para Facturar Venta**: A la hora que llega un cliente este solicita los productos, dicha solicitud es atendida por el vendedor el cual es el encargado de proveer información de los productos en caso del cliente tenga alguna duda o inquietud. Si el producto solicitado por el cliente se encuentra en existencia, el vendedor notifica al cliente sobre el precio del mismo en caso contrario informa al cliente que no se posee ese producto, el vendedor agrega el producto agotado en un lista la cual la entrega al facturador.

Si al cliente no la parece el precio, este solicita descuento, si el descuento es aprobado el facturador procede a realizar el cobro del producto al cliente en caso contrario se termina la venta. Los descuentos solo son aprobados por el propietario. Luego el cliente procede a pagar el producto y el facturador procede a realizarle factura (factura comercial) y a su vez el vendedor empaca el producto (Bolsas plásticas).

Al final de la venta el vendedor entrega el producto al cliente y el facturador registra la venta en el libro diario.

**Control de Inventario:** El facturador en conjunto con los vendedores son los encargados de hacer conteos manuales de las mercancías, para hacerles un informe manual de todos los productos faltantes e informar a la dueña y hacer pedidos a los proveedores. Lo cual provoca que muchas veces los productos se agoten en la tienda y pase mucho tiempo para su reposición, causando perdidas en las tienda por escasez de productos. La dueña revisa la lista de productos agotados y procede ha realizar el pedido a los proveedores.

En caso que el producto este en existencia, el vendedor procede a la clasificación de los mismos (Productos en promoción, descuento, precio normal), luego el Facturador verifica la clasificación, si todo esta bien el propietario calcula el precio de venta de todos los productos y pasa al Facturador la lista de los productos con sus precios de venta.

Después que el Facturador recibe el listado del proveedor, este procede a guardar toda la información en el libro diario y a su vez informa al vendedor los precios para que proceda a etiquetar los productos y luego colocarlos en los estantes de venta.

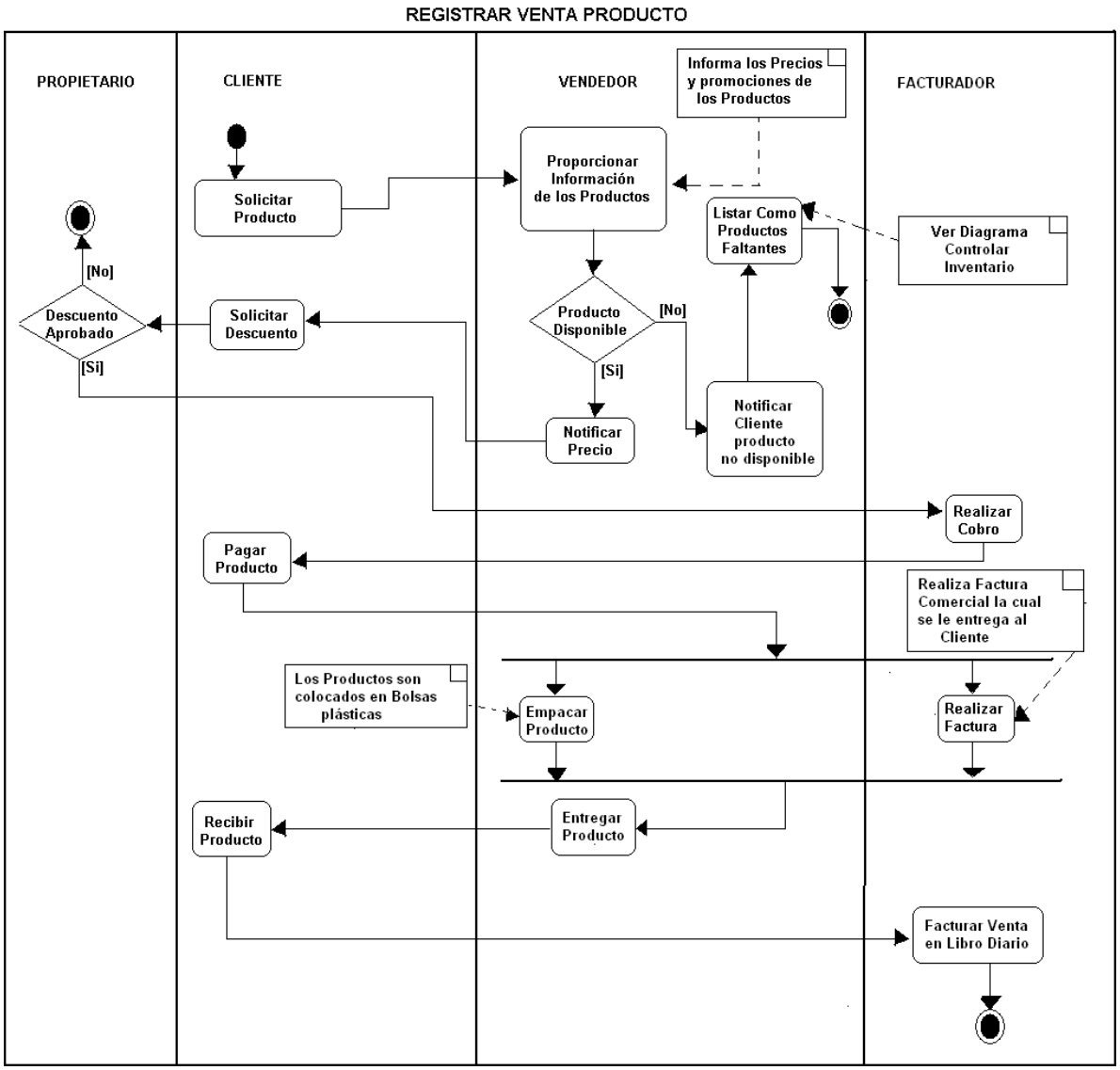
Las promociones se dan de acuerdo a la temporada que nos encontramos y al nivel de demanda de los productos.Todos los datos relevantes de la tienda son anotados en el libro de diario, lo cual a la hora de buscar información en el mismo se hace muy tedioso y cansado. Lo cual provoca mucha perdida de tiempo en tareas que no lo necesitan.

Los precios de ventas son calculados por el propietario de la tienda, el cual aplica el 15% de I.V.A sobre cada producto, después pasa un listado con los productos y sus precios de ventas para su debida marcación (Colocación de etiquetas), el propietario es el que da la autorización de cuales son los productos que están en promoción o en descuento.

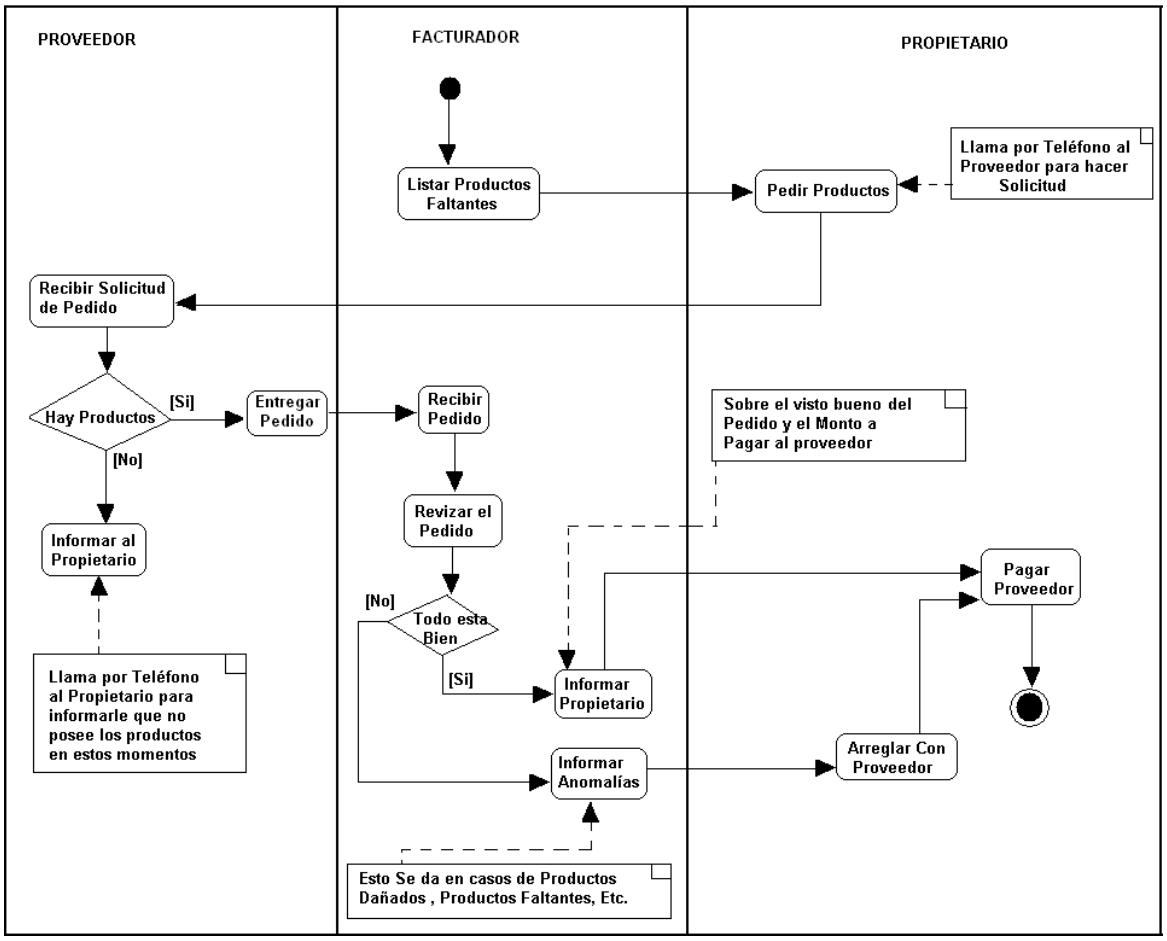
**Arqueo Diario:** El Facturador se encarga de sumar todas las ventas del día luego cuenta el dinero que hay en caja. Si Cuadran ambas cantidades, anota en el libro el monto total de la venta del día. Luego entrega todo el dinero y la lista de la venta del día al propietario. En caso contrario el Facturador revisa detalladamente cada factura, para detectar algún error en las mismas, si no lo hay y sigue faltando dinero, la diferencia entre caja y el monto total de la factura es descalfada del salario del Facturador. El Propietario cuenta el dinero y si cuadra con el informe detallado entregado por el Facturador proceden a cerrar la tienda.

Un dato importante que hay que recalcar es que la tienda no posee una organización bien definida porque solo la integran: la dueña de la misma, el encargado de facturación y dos vendedores, los cuales no poseen sus funciones bien definidas, es decir, hace de todo un poco.

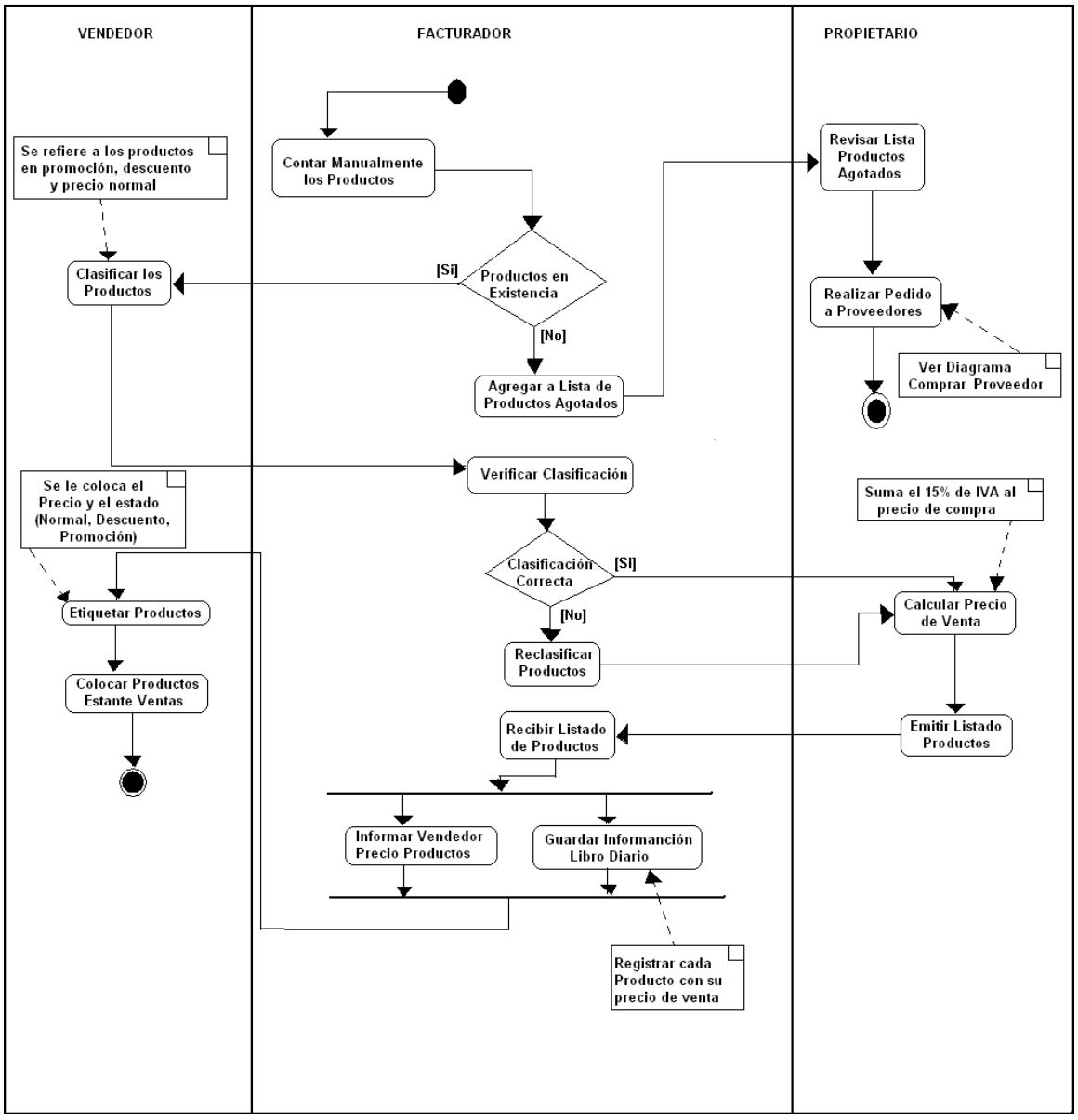
## Diagramas de Actividades



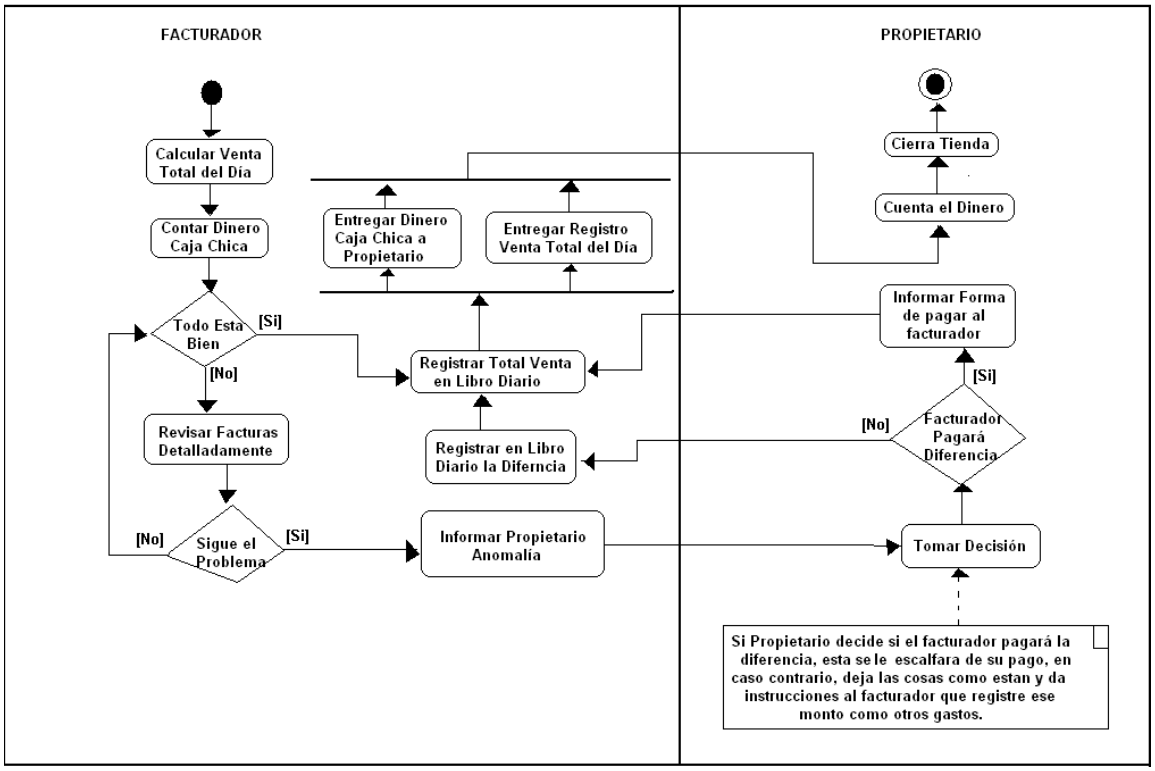
COMPRAR PROVEEDOR



CONTROLAR INVENTARIO



ARQUEO DIARIO



# 4 DESCRIPCION DEL SISTEMA INFORMATICO

El sistema de facturación y control de inventario que se elaborará a la tienda DENMAR deberá ser capaz de hacer cuatro funciones principales:

- ✓ Llevar un control de todas las ventas diarias que se hagan en la tienda, a su vez podrá emitir las facturas que se le entregaran a los clientes después de que realicen cualquier compra de productos, podrá emitir una serie de listados (Productos Más Vendidos, Productos Menos Vendidos, Productos Agotados, Existencia Total de Productos), los cuales se generaran por medio de consultas al sistema. El sistema también podrá administrar las facturas, esto se refiere a los métodos generales en todo sistema: Nuevo, Consultar, Modificar y Eliminar.

- ✓ Otra función que hará el sistema es llevar un control de las compras hecha a los proveedores, emitir recibo a los proveedores. Otra utilidad será la administración de los recibos a los proveedores y los proveedores.

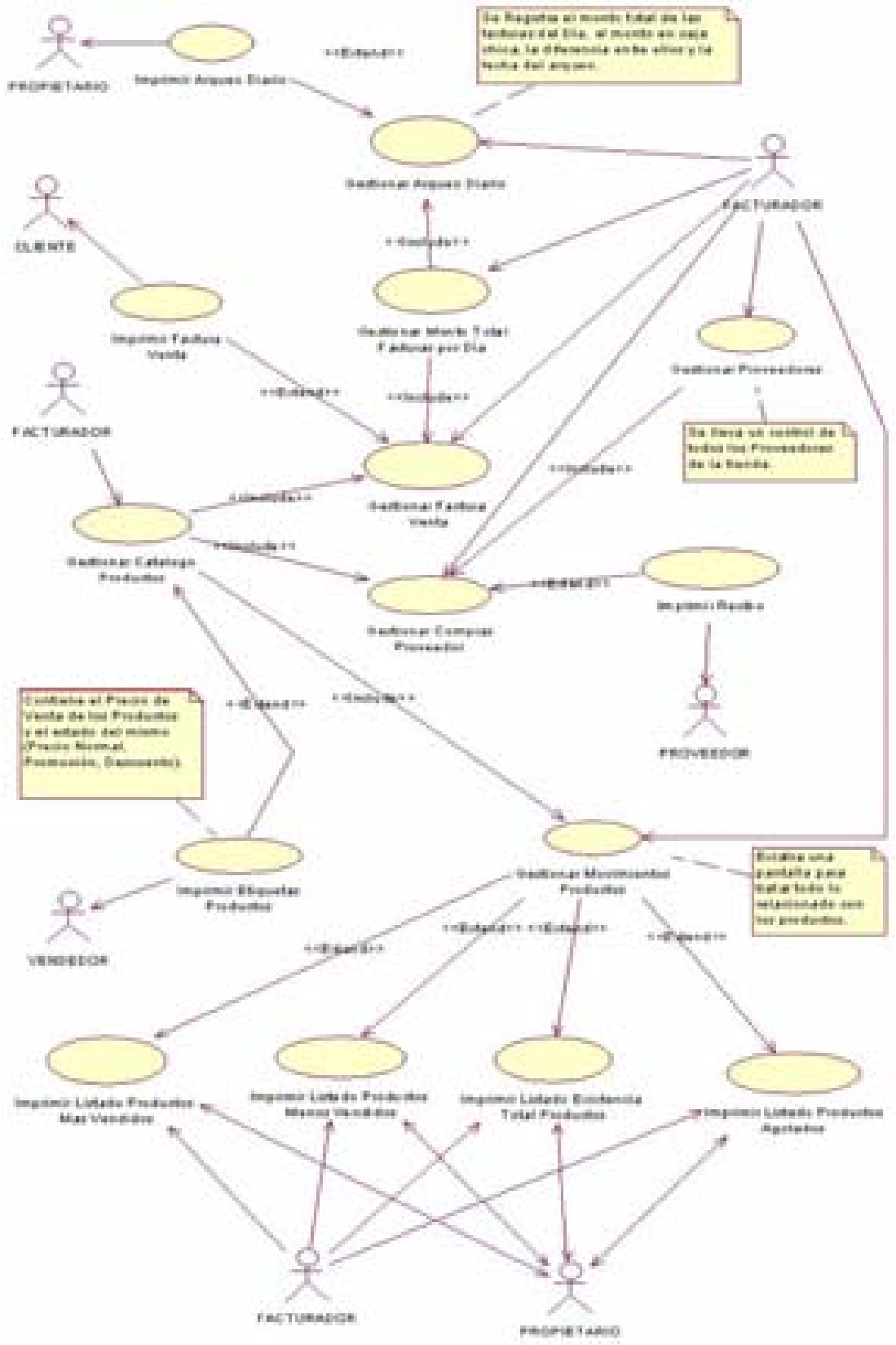
- ✓ Podrá realizar un control de inventario, lo cual se hará por medio de la actualización de los catálogos de los productos.

- ✓ Como última función principal podrá calcular los ingresos de las ventas por día, con lo cual se podrá hacer un arqueo diario con caja chica, para así emitir un informe de Arqueo diario de la tienda, el cual se le entregara al propietario.

Lo antes descrito son las funciones mas principales que efectuara el Sistema de Facturación y Control de Inventario de la tienda DENMAR.

# II ANALISIS

# 1. Diagrama de Casos de uso



**Listado Casos de Uso**

1. Gestionar Arqueo Diario
2. Imprimir Arqueo Diario (Extensión CU.1)
3. Gestionar Monto Total Factura Del Día (Inclusión CU.1), (Inclusión CU.5)
4. Gestionar Proveedores (Inclusión CU.10)
5. Gestionar Factura Venta
6. Imprimir Factura (Extensión CU.5)
7. Gestionar Catalogo Productos (Inclusión CU.5), (Inclusión CU.9), (Inclusión CU.10)
8. Imprimir Etiquetas Productos (Extensión CU.7)
9. Gestionar Movimientos Productos
10. Gestionar Compras Proveedor
11. Imprimir Recibo (Extensión CU.10)
12. Imprimir Listado Productos Mas Vendidos (Extensión CU.9)
13. Imprimir Listado Productos Menos Vendidos (Extensión CU.9)
14. Imprimir Listado Productos Agotados (Extensión CU.9)
15. Imprimir Listado Total Existencia Producto (Extensión CU.9)

## 2. Descripción de Casos de Uso

| **CASO DE USO 1** : | **Gestionar Arqueo Diario** | | |
|---|---|---|---|
| DEFINICIÓN : | Permite Llevar un control de los arqueos diarios de la facturas de ventas y el dinero en caja chica, lo cuales se comparan para verificar si no hay anomalías (No cuadra el total de factura con el monto en caja chica). | | |
| PRIORIDAD : | ○ (1) Vital | ◉ (2) Importante | ○ (3) Conveniente |
| URGENCIA : | ○ (1)Inmediata | ◉ (2) Necesario | ○ (3) Puede Esperar |
| **ACTORES** | | | |
| NOMBRE | DEFINICION | | |
| FACTURADOR | Gestionar ventas, compras de productos. | | |
| **ESCENARIO 1.1** | | | |
| Nombre **:** | **Registrar Arqueo Normalmente** | | |
| Pre-Condiciones **:** | Se realice alguna venta en el día, es decir, hay entrada de efectivo. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | Se hayan facturado todas las ventas hechas en el día. | | |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú arqueos.<br>✓ Se esta dentro de la pantalla Arqueos.<br>✓ Se registra el código y la fecha del arqueo.<br>✓ Se realiza un consulta de las facturas del día por medio de la Fecha de la factura.<br>✓ Se calcula el monto total de las facturas del día.<br>✓ Se registrar el monto en caja chica<br>✓ Se presiona el botón calcular diferencia.(Diferencia entre Total facturas y Caja Chica).<br>✓ Se presiona Guardar arqueo (Fecha Arqueo, Monto Total Facturas del Día, Monto Caja Chica y Diferencia)<br>✓ Se cierra la pantalla. | | |
| Excepciones **:** | ✓ Ya exista el arqueo que se quiere registrar<br>✓ No existan ventas a esa fecha | | |

| ESCENARIO 1.2 | |
|---|---|
| Nombre : | **Registrar Arqueo Con Anomalía** |
| Pre Condiciones : | No se realizo venta en el día, es decir, no hay entrada de efectivo. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | No se haya facturado ninguna venta en el día |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú arqueos.<br>✓ Se debe de estar dentro de la pantalla Arqueos.<br>✓ Se registra el código y la fecha del arqueo.<br>✓ Se calcula el monto total de las facturas del día, lo cual no se Puede por que no se realizo ninguna venta en el día.<br>✓ Se cierra la pantalla. |
| Excepciones : | ✓ Registro incompleto |
| **ESCENARIO 1.3** | |
| Nombre : | **Consultar Arqueo Normalmente** |
| Pre-Condiciones : | Se debe de estar dentro de la pantalla Arqueos. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Exista un arqueo registrado en el sistema |
| Operaciones : | ✓ Se selecciona el menú mostrar y después el submenú arqueos.<br>✓ Se presiona el botón buscar arqueo.<br>✓ Se introduce el parámetro de consulta Correctamente (Fecha).<br>✓ Se busca el arqueo interesado en la respuesta de la consulta.<br>✓ Se cierra la pantalla. |
| Excepciones : | ✓ No exista el arqueo |
| **ESCENARIO 1.4** | |
| Nombre : | **Consultar Arqueo Con Parámetro Incorrecto** |
| Pre-Condiciones : | Exista un arqueo registrado en el sistema |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Exista un arqueo registrado en el sistema |
| Operaciones : | ✓ Se selecciona el menú mostrar y después el submenú arqueos.<br>✓ Se presiona el botón buscar arqueo.<br>✓ Se introduce el parámetro de consulta Incorrecto.<br>✓ No hay arqueos con ese parámetro introducido.<br>✓ Se cierra la pantalla. |

| | |
|---|---|
| Excepciones **:** | ✓ No exista el arqueo |
| **ESCENARIO 1.5** | |
| Nombre **:** | **Modificar Arqueo** |
| Pre-Condiciones **:** | Exista un arqueo registrado en el sistema, al cual se le va a realizar una variación. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú arqueos.<br>✓ Se debe de estar dentro de la pantalla Arqueos.<br>✓ Se digita todos los datos del arqueo con las debidas modificaciones.<br>✓ Se presiona el botón modificar arqueo.<br>✓ Se cierra la pantalla |
| Excepciones **:** | ✓ No exista el arqueo |
| **ESCENARIO 1.6** | |
| Nombre **:** | **Anular Arqueo** |
| Pre-Condiciones **:** | Exista un arqueo registrado en el sistema y que este mal registrado. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú arqueos.<br>✓ Se debe de estar dentro de la pantalla Arqueos.<br>✓ Se digita el código del arqueo que se desea eliminar.<br>✓ Se presiona el botón eliminar arqueo.<br>✓ Se cierra la pantalla |
| Excepciones **:** | ✓ No exista el arqueo<br>✓ El arqueo ya fue anulado |

| **CASO DE USO 3** : | **Gestionar Monto Total Factura por Día** | | |
|---|---|---|---|
| DEFINICIÓN : | Se consulta al sistema las facturas de venta del día por medio del parámetro fecha, se calcula el total de las facturas. | | |
| PRIORIDAD : | ○ (1) Vital | ◉ 2) Importante | ○ (3) Conveniente |
| URGENCIA : | ○ (1) Inmediata | ◉ (2) Necesario | ○ (3) Puede Esperar |
| **ACTORES** | | | |
| NOMBRE | DEFINICION | | |
| FACTURADOR | Gestionar ventas, compras de productos. | | |
| **ESCENARIO 3.1** | | | |
| Nombre **:** | **Registrar Cálculos** | | |
| Pre-Condiciones **:** | Existan ventas registradas en el día. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | El facturador solicite los cálculos al sistema. | | |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú arqueos.<br>✓ Se debe de estar dentro de la pantalla Arqueos.<br>✓ Se Introduce código y fecha del arqueo<br>✓ Se elije la opción consultar.<br>✓ Se introduce el parámetro de consulta correcto. (fecha)<br>✓ Se calcula el total de todas las facturas consultadas.<br>✓ Se Presiona Guardar Arqueo.<br>✓ Se cierra la pantalla. | | |
| Excepciones **:** | ✓ Ya existan los cálculos en la base de datos<br>✓ No existan facturas de ventas en esa fecha | | |
| **ESCENARIO 3.2** | | | |
| Nombre **:** | **Consultar Cálculos** | | |
| Pre-Condiciones **:** | Existan cálculos generados. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | Existan cálculos registrados en el sistema. | | |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú arqueos.<br>✓ Se debe de estar dentro de la pantalla Arqueos.<br>✓ Se presiona el botón buscar arqueo.<br>✓ Se introduce el parámetro de consulta correcto. (Fecha)<br>✓ Se cierra la pantalla. | | |
| Excepciones **:** | ✓ No existan los cálculos | | |

| **CASO DE USO 4** : | **Gestionar Proveedores** | | |
|---|---|---|---|
| DEFINICIÓN : | Se lleva un control en el sistema de los datos más relevantes de los proveedores de la tienda. | | |
| PRIORIDAD : | (1) Vital | (2) Importante | (3) Conveniente |
| URGENCIA : | (1)Inmediata | (2) Necesario | (3) Puede Esperar |
| **ACTORES** | | | |
| NOMBRE | DEFINICION | | |
| FACTURADOR | Gestionar ventas, compras de productos. | | |
| **ESCENARIO 4.1** | | | |
| Nombre **:** | **Registrar Proveedor** | | |
| Pre-Condiciones **:** | El proveedor no este registrado en el sistema. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | | | |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú proveedor.<br>✓ Se debe de estar dentro de la pantalla Proveedor.<br>✓ Se digita los datos del proveedor.<br>✓ Se presiona el botón agregar fila.<br>✓ Se presiona guardar proveedor.<br>✓ Se cierra pantalla. | | |
| Excepciones **:** | ✓ Ya exista el proveedor en la base de datos | | |
| **ESCENARIO 4.2** | | | |
| Nombre **:** | **Consultar Proveedor** | | |
| Pre-Condiciones **:** | El proveedor este registrado previamente. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | Existan datos de proveedores en el sistema. | | |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú proveedor.<br>✓ Se debe de estar dentro de la pantalla Proveedor.<br>✓ Se presiona el botón buscar proveedor.<br>✓ Se introduce los parámetros de la consulta (Id Proveedor)<br>✓ Se cierra pantalla. | | |
| Excepciones **:** | ✓ No exista el proveedor en la base de datos<br>✓ Se introduce mal el código del proveedor | | |

| ESCENARIO 4.3 | |
|---|---|
| Nombre **:** | **Modificar Proveedor** |
| Pre-Condiciones **:** | El proveedor este registrado. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Existan datos de proveedores en el sistema. |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú proveedor.<br>✓ Se debe de estar dentro de la pantalla Proveedor.<br>✓ Se digita los datos del proveedor con las respectivas modificaciones.<br>✓ Se presiona el botón modificar proveedor.<br>✓ Se cierra pantalla. |
| Excepciones **:** | ✓ No exista el proveedor en la base de datos<br>✓ Se introduce mal el código del proveedor |
| **ESCENARIO 4.4** | |
| Nombre **:** | **Eliminar Proveedor** |
| Pre-Condiciones **:** | El proveedor este mal registrado. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Existan datos de proveedores en el sistema. |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú proveedor.<br>✓ Se debe de estar dentro de la pantalla Proveedor.<br>✓ Se digita el id del proveedor a eliminar.<br>✓ Se presiona el botón eliminar proveedor.<br>✓ Se cierra pantalla. |
| Excepciones **:** | ✓ No exista el proveedor en la base de datos<br>✓ El proveedor ya fue eliminado |

| **CASO DE USO 5** : | **Gestionar Factura Venta** | | |
|---|---|---|---|
| DEFINICIÓN : | Permite Llevar un control de todas las ventas realizadas en el día. | | |
| PRIORIDAD : | (1) Vital | (2) Importante | (3) Conveniente |
| URGENCIA : | (1)Inmediata | (2) Necesario | (3) Puede Esperar |
| **ACTORES** | | | |
| NOMBRE | DEFINICION | | |
| FACTURADOR | Gestionar ventas, compras de productos. | | |
| **ESCENARIO 5.1** | | | |
| Nombre **:** | **Registrar Factura de Venta Normalmente** | | |
| Pre-Condiciones **:** | Se realiza una venta en la tienda. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | Hay productos solicitados en existencia | | |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú facturar venta.<br>✓ Se debe de estar dentro de la pantalla Factura Venta.<br>✓ En la parte de facturar venta se introduce el código de la factura, la fecha, le descripción y el nombre del cliente.<br>✓ En la parte del detalle de venta se introduce el código de la factura, el código del producto y la cantidad.<br>✓ El sistema verifica la existencia del producto.<br>✓ El sistema rellena el precio automáticamente que se introduce el código del producto.<br>✓ Se presiona el botón calcular en la parte del detalle de venta.<br>✓ Se presiona el botón calcular total factura.<br>✓ Se presiona el botón agregar fila en la parte del detalle de venta.<br>✓ Se digita el monto pagado.<br>✓ Se presiona el botón calcular vuelto.<br>✓ Se presiona el botón agregar fila en la parte de facturar venta.<br>✓ Se presiona el botón Guardar Factura.<br>✓ Se cierra la pantalla. | | |
| Excepciones **:** | ✓ La factura ya fue registrada | | |

| ESCENARIO 5.2 | |
|---|---|
| Nombre **:** | **Registrar Factura de Venta Anomalía** |
| Pre-Condiciones **:** | Se realiza una venta en la tienda. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | No hay productos solicitados en existencia |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú facturar venta.<br>✓ Se debe de estar dentro de la pantalla Factura Venta.<br>✓ En la parte de facturar venta se introduce el código de la factura, la fecha, le descripción y el nombre del cliente.<br>✓ En la parte del detalle de venta se introduce el código de la factura, el código del producto y la cantidad.<br>✓ El sistema verifica la existencia del producto. (El producto esta agotado o la cantidad solicitada excede la cantidad en existencia).<br>✓ Cerrar Ventana. |
| Excepciones **:** | ✓ La compra de productos no esta registrada aun en la base de datos. |
| **ESCENARIO 5.3** | |
| Nombre **:** | **Consultar Factura Venta Normalmente** |
| Pre-Condiciones **:** | Se encuentre la factura de venta registrada en el sistema. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Existan facturas en la base de datos |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú facturar venta.<br>✓ Se debe de estar dentro de la pantalla Factura Venta<br>✓ Se presiona el botón buscar factura.<br>✓ Se introduce el parámetro de consulta Correctamente (Código de la factura).<br>✓ Se presiona el botón mostrar detalle de factura.<br>✓ Se cierra la pantalla. |
| Excepciones **:** | ✓ No existan facturas en la base de datos |

| ESCENARIO 5.4 | |
|---|---|
| Nombre **:** | **Consultar Factura Venta con Mal Parámetro** |
| Pre-Condiciones **:** | Se encuentre la factura de venta registrada en el sistema. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Se encuentre la factura de venta registrada en el sistema. |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú facturar venta.<br>✓ Se debe de estar dentro de la pantalla Factura Venta<br>✓ Se presiona el botón buscar factura.<br>✓ Se introduce el parámetro de consulta incorrecto (Código de la factura).<br>✓ Se cierra la pantalla. |
| Excepciones **:** | ✓ No existan facturas en la base de datos |
| **ESCENARIO 5.5** | |
| Nombre **:** | **Anular Factura Venta** |
| Pre-Condiciones **:** | Se encuentre la factura de venta mal registrada. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Se encuentre la factura de venta registrada en el sistema. |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú facturar venta.<br>✓ Se debe de estar dentro de la pantalla Factura Venta<br>✓ Se introduce el código de la factura a eliminar<br>✓ Se presiona el botón anular factura.<br>✓ Se cierra la pantalla |
| Excepciones **:** | ✓ No existan facturas en la base de datos<br>✓ La factura ya fue anulada |

| **CASO DE USO 7** : | **Gestionar Catalogo Productos** | | |
|---|---|---|---|
| DEFINICIÓN : | Nos sirve para ver el estado que se encuentra el producto (Existente, Agotado, Dañado) | | |
| PRIORIDAD : | (1) Vital | (2) Importante | (3) Conveniente |
| URGENCIA : | (1)Inmediata | (2) Necesario | (3) Puede Esperar |
| **ACTORES** | | | |
| NOMBRE | DEFINICION | | |
| FACTURADOR | Gestionar ventas, compras de productos. | | |
| **ESCENARIO 7.1** | | | |
| Nombre **:** | **Consultar Productos** | | |
| Pre-Condiciones **:** | El producto este registrado en el sistema. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | Haya registros en la base de datos | | |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú producto.<br>✓ Se debe de estar dentro de la pantalla Productos<br>✓ Se presiona el botón buscar producto.<br>✓ Se introduce el parámetro de consulta (código producto)<br>✓ Verifica Datos<br>✓ Se cierra Ventana | | |
| Excepciones **:** | ✓ No exista el producto en la base de datos | | |
| **ESCENARIO 7.2** | | | |
| Nombre **:** | **Consultar Categoría** | | |
| Pre-Condiciones **:** | La categoría esta registrada en el sistema. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | Haya registros en la base de datos | | |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú categoría producto.<br>✓ Se debe de estar dentro de la pantalla categoría producto<br>✓ Se presiona el botón buscar categoría.<br>✓ Se introduce el parámetro de consulta (código categoría)<br>✓ Verifica Datos<br>✓ Se cierra Ventana | | |
| Excepciones **:** | ✓ No exista la categoría en la base de datos | | |

| ESCENARIO 7.3 | |
|---|---|
| Nombre **:** | **Consultar Subcategoría** |
| Pre-Condiciones **:** | La subcategoría esta registrada en el sistema. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Haya registros en la base de datos |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú subcategoría producto.<br>✓ Se debe de estar dentro de la pantalla subcategoría producto<br>✓ Se presiona el botón buscar subcategoría.<br>✓ Se introduce el parámetro de consulta (código subcategoría)<br>✓ Verifica Datos<br>✓ Se cierra Ventana |
| Excepciones **:** | ✓ No exista la subcategoría en la base de datos |

| **CASO DE USO 9** : | **Gestionar Movimientos Productos** | | |
|---|---|---|---|
| DEFINICIÓN : | Aquí es donde se realizara el control de inventario, es decir después de cada venta, compra y producto dañado, el Facturador tendrá que entrar en este pantalla para actualizar el catalogo de los productos. El producto se categoriza como dañado una vez que el vendedor informa al propietario quien autoriza al facturador el cambio de estado del producto dentro del sistema, esto se da en la pantalla producto (modificar cantidad producto). | | |
| PRIORIDAD : | (1) Vital | (2) Importante | (3) Conveniente |
| URGENCIA : | (1)Inmediata | (2) Necesario | (3) Puede Esperar |
| **ACTORES** | | | |
| NOMBRE | DEFINICION | | |
| FACTURADOR | Gestionar ventas, compras de productos. | | |
| **ESCENARIO 9.1** | | | |
| Nombre **:** | **Actualizar Catalogo Después de Venta** | | |
| Pre-Condiciones **:** | Se haya realizado alguna venta a un cliente. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | La venta esta registrada en el sistema. | | |
| Operaciones **:** | ✓ Después que se siguen los pasos del caso de uso 5 escenario1 (Registrar una factura de venta normalmente).<br>✓ El sistema actualiza automáticamente el inventario.<br>✓ Cierre de pantalla | | |
| Excepciones **:** | ✓ Haya conflicto al registra la venta | | |
| **ESCENARIO 9.2** | | | |
| Nombre **:** | **Actualizar Catalogo Después de Comprar** | | |
| Pre-Condiciones | Se haya realizado alguna compra a algún proveedor. | | |
| Iniciado por **:** | FACTURADOR | | |
| Finalizado por **:** | SISTEMA | | |
| Post-Condiciones **:** | La compra esta registrada en el sistema. | | |
| Operaciones **:** | ✓ Después que se siguen los pasos del caso de uso 10 escenario1 (Registrar una factura de compra normalmente).<br>✓ El sistema actualiza automáticamente el inventario.<br>✓ Cierre de pantalla | | |
| Excepciones **:** | ✓ Haya conflicto al registra la compra | | |

| ESCENARIO 9.3 | |
|---|---|
| Nombre : | **Registrar nueva categoría** |
| Pre-Condiciones | La categoría no esta registrada en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Se adquieran nuevos productos en la tienda o se cambie de categoría un producto. |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú categoría producto.<br>✓ Se debe de estar dentro de la pantalla categoría producto<br>✓ Se introduce los datos de la categoría<br>✓ Se presiona el botón agregar fila.<br>✓ Se presiona el botón guardar categoría.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ La categoría ya este registrada |
| **ESCENARIO 9.4** | |
| Nombre : | **Registrar nueva subcategoría** |
| Pre-Condiciones | La subcategoría no esta registrada en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Se adquieran nuevos productos en la tienda o se cambie de categoría un producto. |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú subcategoría producto.<br>✓ Se debe de estar dentro de la pantalla subcategoría producto<br>✓ Se introduce los datos de la subcategoría<br>✓ Se presiona el botón agregar fila.<br>✓ Se presiona el botón guardar subcategoría.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ La subcategoria ya este registrada |

| ESCENARIO 9.5 | |
|---|---|
| Nombre : | **Registrar nuevo producto** |
| Pre-Condiciones : | El producto no este registrado en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Se adquieran nuevos productos en la tienda o se cambie de categoría un producto. |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú producto.<br>✓ Se debe de estar dentro de la pantalla producto<br>✓ Se introduce los datos del producto<br>✓ Se presiona el botón agregar fila.<br>✓ Se presiona el botón guardar producto.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ El producto ya este registrado |
| **ESCENARIO 9.6** | |
| Nombre : | **Modificar Categoría** |
| Pre-Condiciones : | La categoría este mal registrada en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Existan datos en la base de datos |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú categoría producto.<br>✓ Se debe de estar dentro de la pantalla categoría<br>✓ Se introduce los datos de la categoría con las modificaciones deseadas.<br>✓ Se presiona el botón modificar categoría.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ Exista la categoría en la base de datos |
| **ESCENARIO 9.7** | |
| Nombre : | **Modificar Subcategoría** |
| Pre-Condiciones : | La subcategoría este mal registrada en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Existan datos en la base de datos |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú subcategoría producto.<br>✓ Se debe de estar dentro de la pantalla subcategoría<br>✓ Se introduce los datos de la subcategoría con las modificaciones deseadas.<br>✓ Se presiona el botón modificar subcategoría.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ Exista la subcategoría en la base de datos |

| ESCENARIO 9.8 | |
|---|---|
| Nombre : | **Modificar Producto** |
| Pre-Condiciones : | El producto este mal registrado en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Existan datos en la base de datos |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú producto.<br>✓ Se debe de estar dentro de la pantalla producto.<br>✓ Se introduce los datos del producto con las modificaciones deseadas.<br>✓ Se presiona el botón modificar producto.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ Exista el producto en la base de datos |
| **ESCENARIO 9.9** | |
| Nombre : | **Eliminar Categoría** |
| Pre-Condiciones : | La categoría este mal registrada en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Existan datos en la base de datos |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú categoría producto.<br>✓ Se debe de estar dentro de la pantalla categoría<br>✓ Se digita el código de la categoría que se desea eliminar.<br>✓ Se presiona el botón eliminar categoría.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ Exista la categoría en la base de datos<br>✓ La categoría ya fue eliminada |
| **ESCENARIO 9.10** | |
| Nombre : | **Eliminar Subcategoría** |
| Pre-Condiciones : | La subcategoría este mal registrada en el sistema. |
| Iniciado por : | FACTURADOR |
| Finalizado por : | SISTEMA |
| Post-Condiciones : | Existan datos en la base de datos |
| Operaciones : | ✓ Se selecciona el menú formularios y después el submenú subcategoría producto.<br>✓ Se debe de estar dentro de la pantalla subcategoría<br>✓ Se digita el código de la subcategoría que se desea eliminar.<br>✓ Se presiona el botón eliminar subcategoría.<br>✓ Se cierra Ventana |
| Excepciones : | ✓ Exista la subcategoría en la base de datos<br>✓ La subcategoría ya fue eliminada |

| ESCENARIO 9.11 | |
|---|---|
| Nombre **:** | **Eliminar Producto** |
| Pre-Condiciones **:** | El Producto este mal registrado en el sistema. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Existan datos en la base de datos |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú producto.<br>✓ Se debe de estar dentro de la pantalla producto.<br>✓ Se digita el código del producto que se desea eliminar.<br>✓ Se presiona el botón eliminar producto.<br>✓ Se cierra Ventana |
| Excepciones **:** | ✓ Exista el producto en la base de datos<br>✓ El producto ya fue eliminado |

| **CASO DE USO 10** : | **Gestionar Compra Proveedor** | | |
|---|---|---|---|
| DEFINICIÓN : | Permite Llevar un control de todas las compras realizadas. | | |
| PRIORIDAD : | (1) Vital | (2) Importante | (3) Conveniente |
| URGENCIA : | (1)Inmediata | (2) Necesario | (3) Puede Esperar |

| **ACTORES** | |
|---|---|
| NOMBRE | DEFINICION |
| FACTURADOR | Gestionar ventas, compras de productos. |
| | |

| ESCENARIO 10.1 | |
|---|---|
| Nombre **:** | **Registrar Factura de Compra Normalmente** |
| Pre-Condiciones **:** | Llegue el pedido hecho al proveedor a la tienda. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | No exista ninguna anomalía en el pedido. |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú facturar compra.<br>✓ Se debe de estar dentro de la pantalla Factura Compra.<br>✓ En la parte de facturar compra se introduce el código de la factura, código proveedor y la fecha.<br>✓ En la parte del detalle de compra se introduce el código de la factura, el código del producto, el producto y la cantidad. |

| | |
|---|---|
| | ✓ El sistema rellena el precio automáticamente que se introduce el código del producto.<br>✓ Se presiona el botón calcular en la parte del detalle de compra.<br>✓ Se presiona el botón calcular total factura.<br>✓ Se presiona el botón agregar fila en la parte del detalle de compra.<br>✓ Se presiona el botón agregar fila en la parte de facturar compra.<br>✓ Se presiona el botón Guardar Factura.<br>✓ Se cierra la pantalla. |
| Excepciones **:** | ✓ La factura ya este registrada |
| **ESCENARIO 10.2** | |
| Nombre **:** | **Registrar Factura de Compra de producto no registrado** |
| Pre-Condiciones **:** | Llegue el pedido hecho al proveedor a la tienda. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | No exista ninguna anomalía en el pedido. |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú facturar compra.<br>✓ Se debe de estar dentro de la pantalla Factura Compra.<br>✓ En la parte de facturar compra se introduce el código de la factura, código proveedor y la fecha.<br>✓ En la parte del detalle de compra se introduce el código de la factura, el código del producto, el producto, la cantidad y el precio.<br>✓ Se presiona el botón calcular en la parte del detalle de compra.<br>✓ Se presiona el botón calcular total factura.<br>✓ Se presiona el botón agregar fila en la parte del detalle de compra.<br>✓ Se presiona el botón agregar fila en la parte de facturar compra.<br>✓ Se presiona el botón Guardar Factura.<br>✓ Se ingresa producto en catalogo (Caso de Uso 9, Escenario 5)<br>✓ Cerrar Pantalla |
| Excepciones **:** | ✓ Haya fallas en la base de datos |

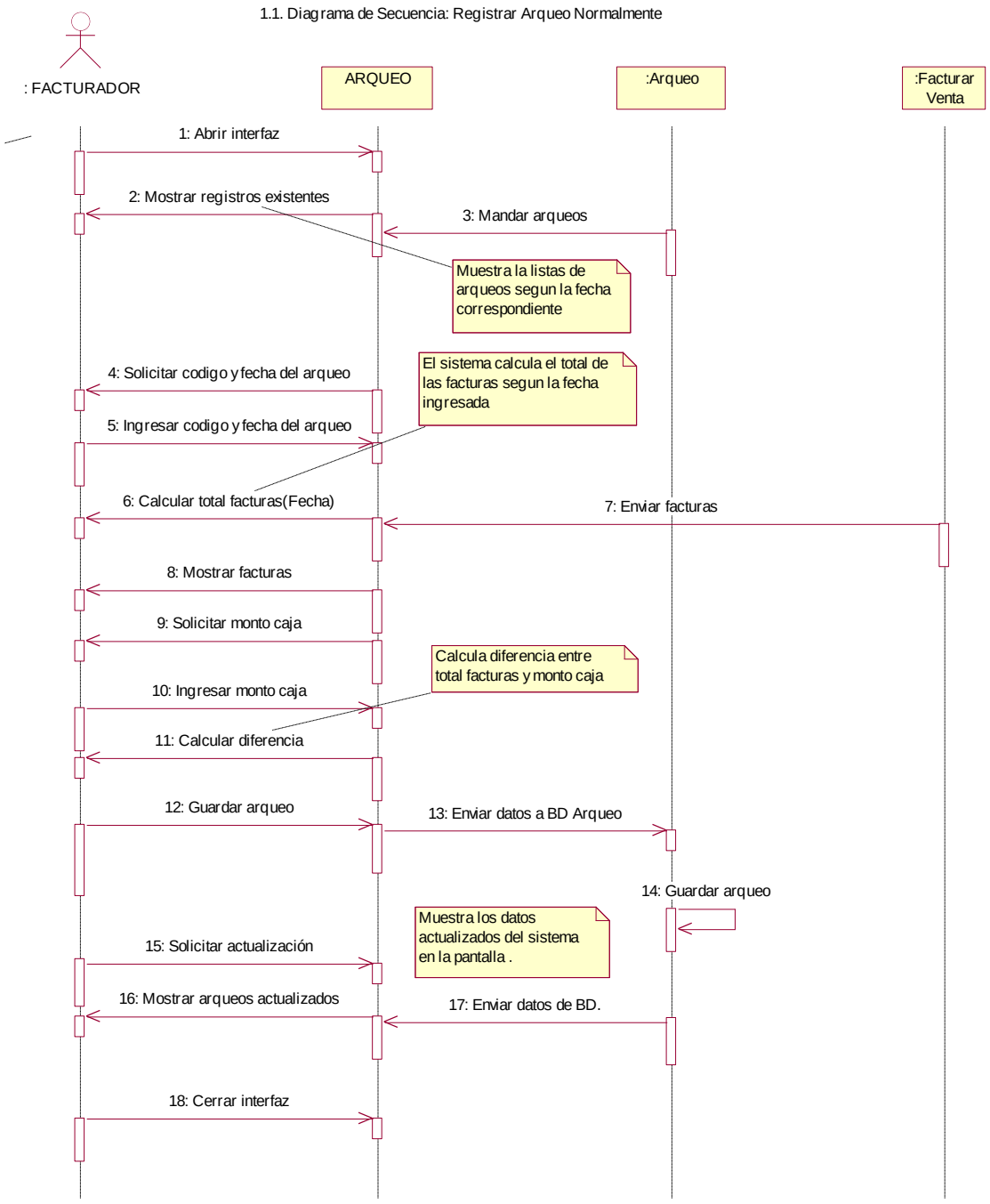
| ESCENARIO 10.3 | |
|---|---|
| Nombre **:** | **Consultar Factura de Compra Normalmente** |
| Pre-Condiciones **:** | Haya alguna factura de compra registrada en el sistema. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Existan facturas en la base de datos |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú facturar compra.<br>✓ Se debe de estar dentro de la pantalla Factura Compra<br>✓ Se presiona el botón buscar factura.<br>✓ Se introduce el parámetro de consulta Correctamente (Código de la factura).<br>✓ Se presiona el botón mostrar detalle de factura.<br>✓ Se cierra la pantalla. |
| Excepciones **:** | ✓ Los datos estén erróneos en la factura |
| **ESCENARIO 10.4** | |
| Nombre **:** | **Consultar Factura Compra con Mal Parámetro** |
| Pre-Condiciones **:** | Haya alguna factura de compra registrada en el sistema. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Existan facturas en la base de datos |
| Operaciones **:** | ✓ Se selecciona el menú mostrar y después el submenú facturar compra.<br>✓ Se debe de estar dentro de la pantalla Factura Compra.<br>✓ Se presiona el botón buscar factura.<br>✓ Se introduce el parámetro de consulta incorrecto (Código de la factura).<br>✓ Se cierra la pantalla. |
| Excepciones **:** | ✓ Los datos estén erróneos en la factura |

| ESCENARIO 10.5 | |
|---|---|
| Nombre **:** | **Anular Factura de Compra** |
| Pre-Condiciones **:** | Haya alguna factura de compra mal registrada en el sistema. |
| Iniciado por **:** | FACTURADOR |
| Finalizado por **:** | SISTEMA |
| Post-Condiciones **:** | Existan facturas en la base de datos |
| Operaciones **:** | ✓ Se selecciona el menú formularios y después el submenú facturar compra.<br>✓ Se debe de estar dentro de la pantalla Factura Compra.<br>✓ Se introduce el código de la factura a eliminar<br>✓ Se presiona el botón anular factura.<br>✓ Se cierra la pantalla |
| Excepciones **:** | ✓ Los datos estén erróneos en la factura<br>✓ La factura ya fue anulada |

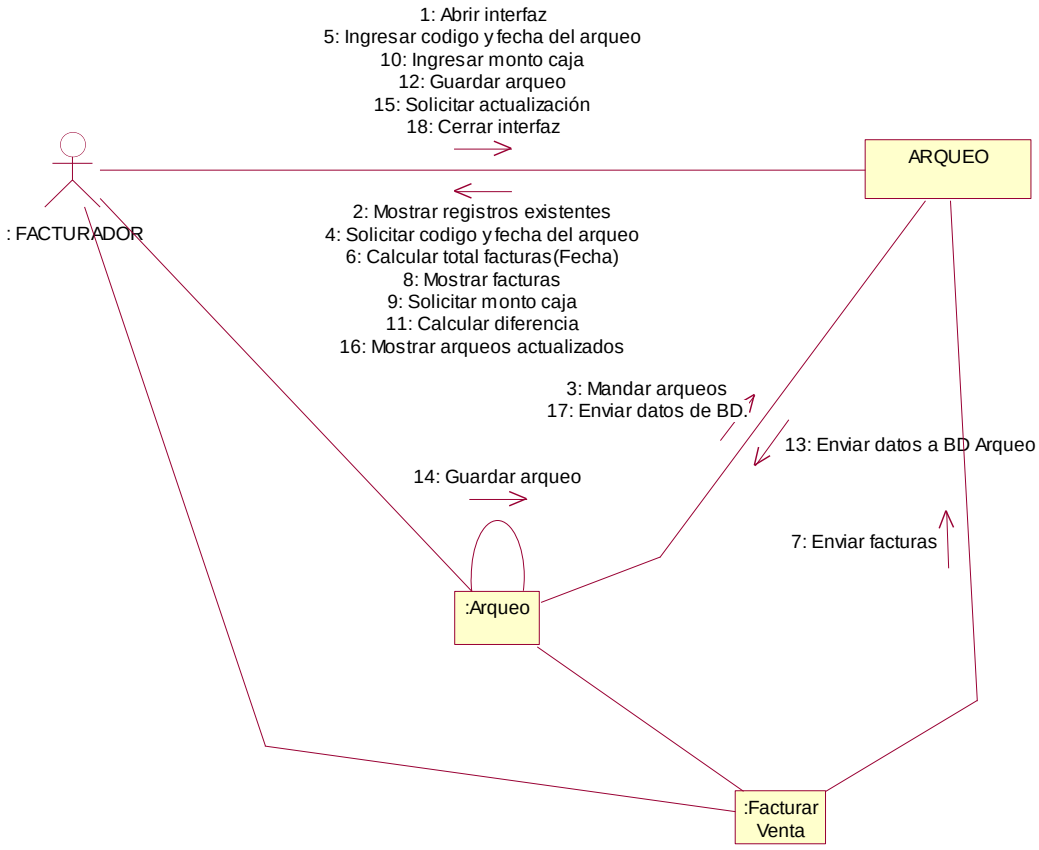
# 3. DIAGRAMAS DE INTERACCION

## Caso de Uso 1: Gestionar Arqueo Diario

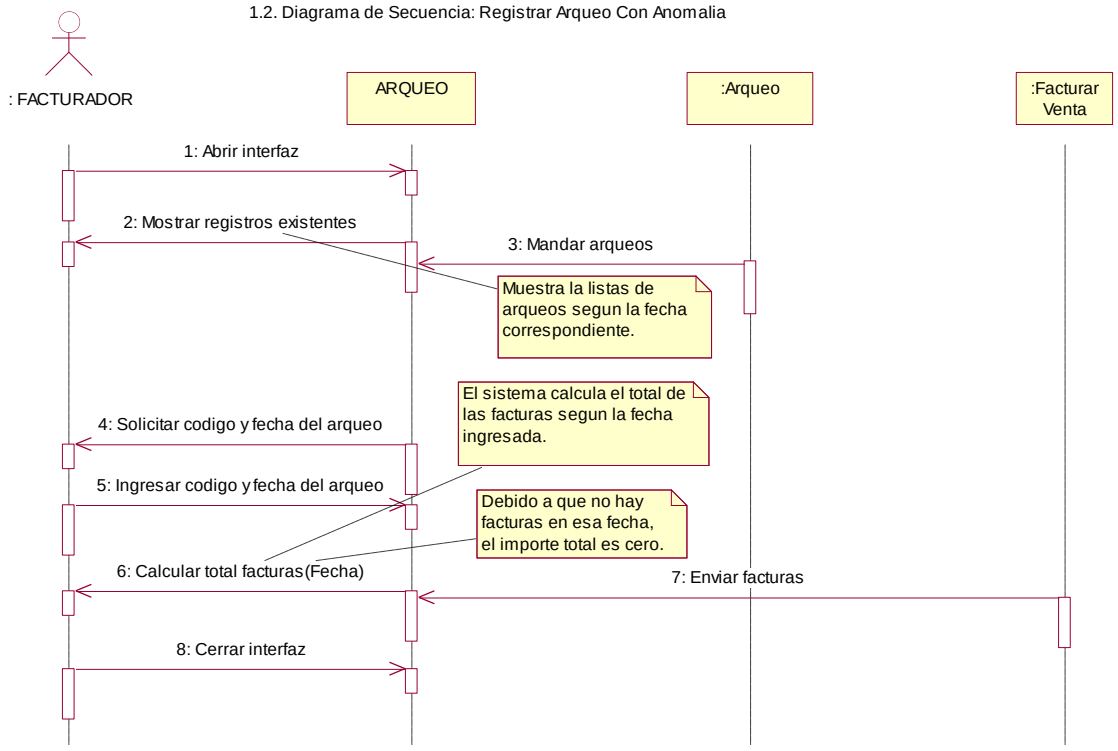
## Escenario 1: Registrar arqueo normalmente.

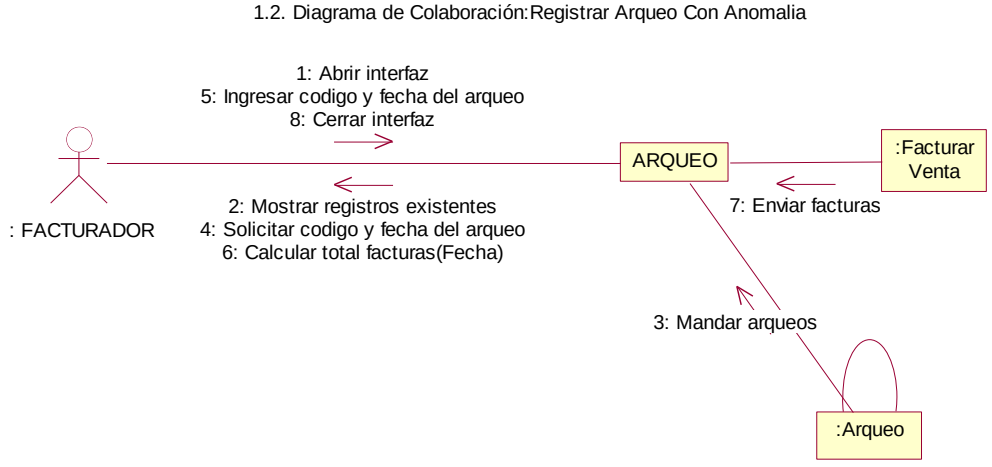


1.1. Diagrama de Colaboración:Registrar Arqueo Normalmente

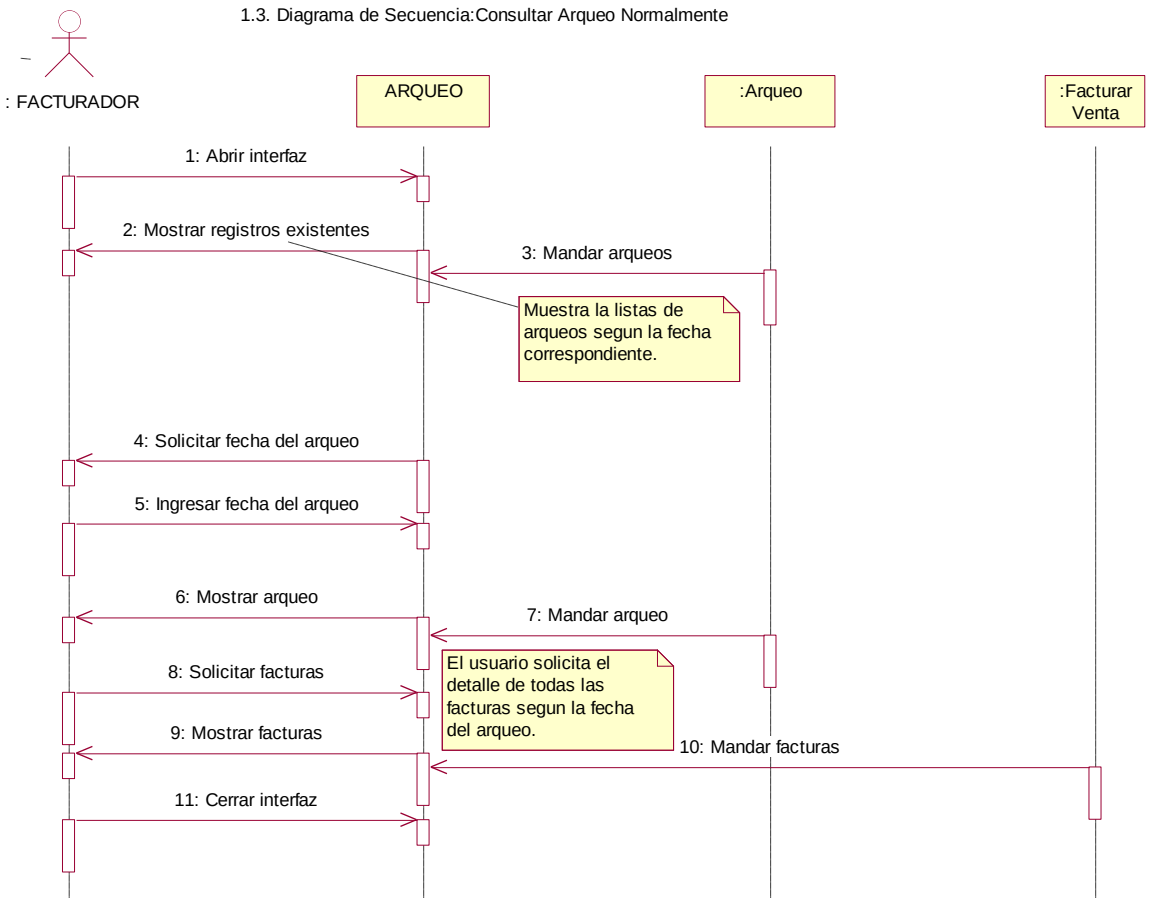


## Escenario 2: Registrar arqueo con anomalía.

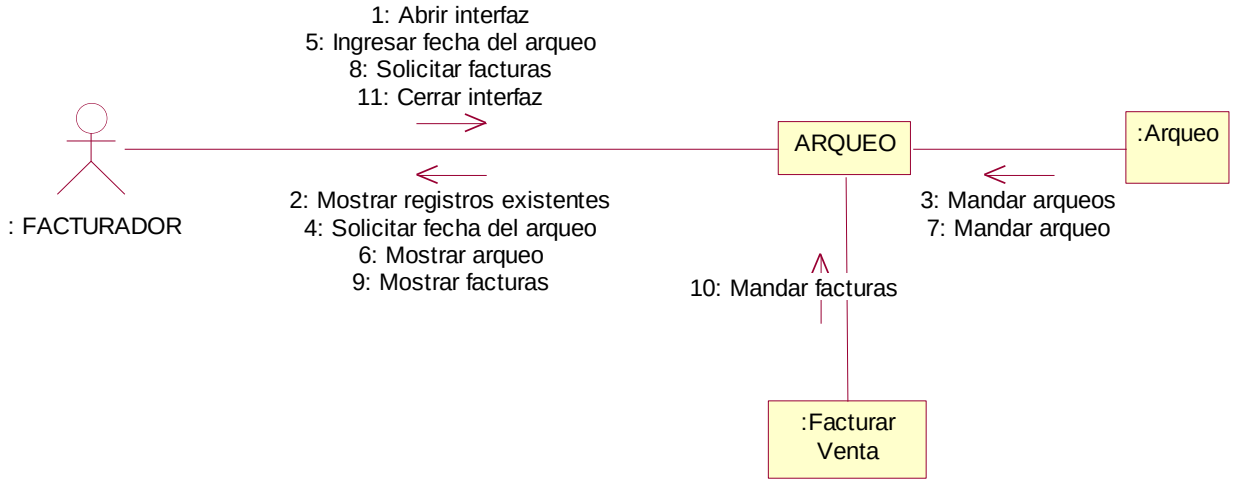


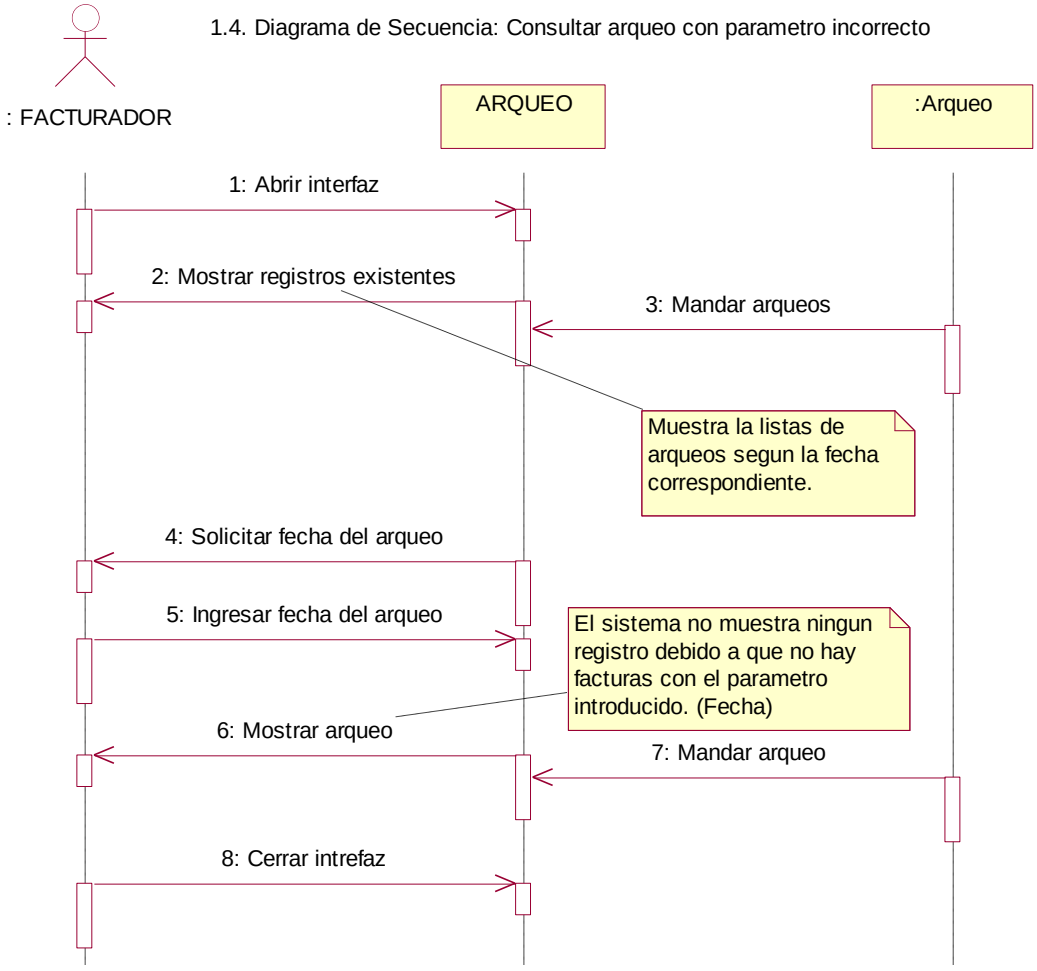
## Escenario 3: Consultar Arqueo Normalmente



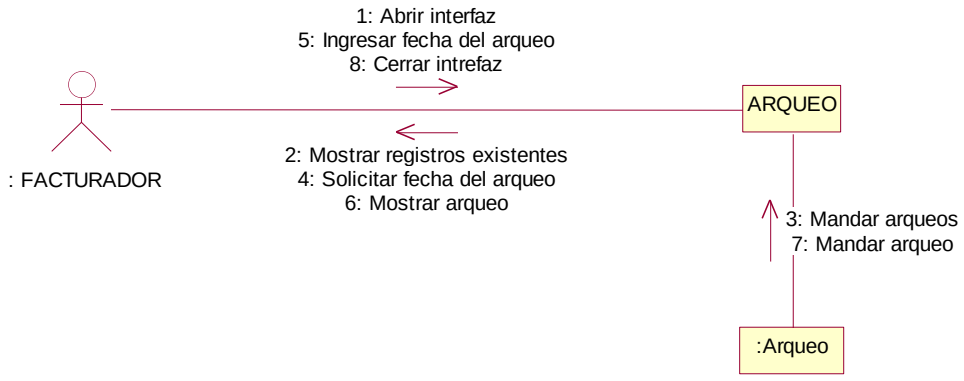

1.3. Diagrama de Colaboración: Consultar Arqueo Normalmente



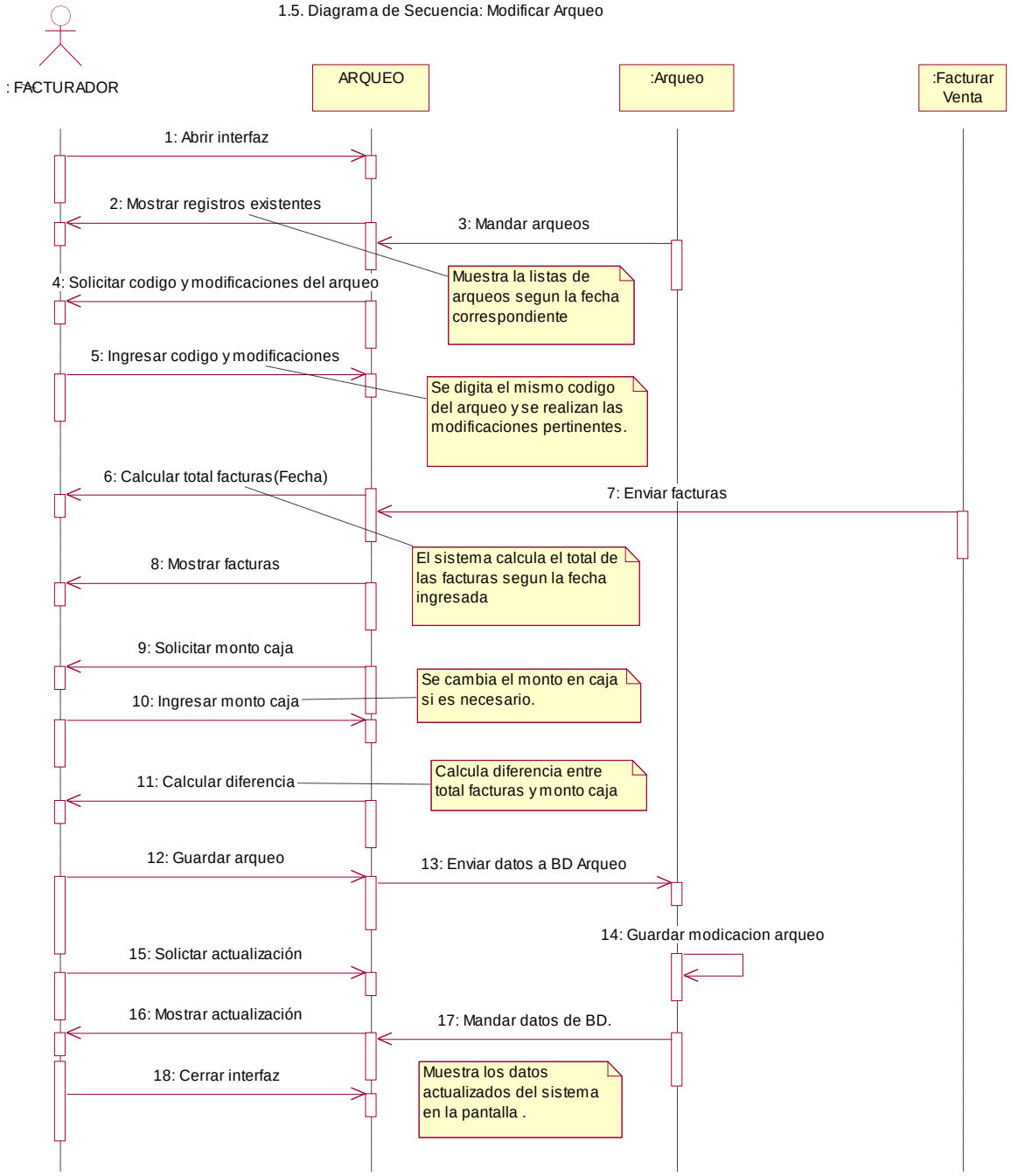
## Escenario 4: Consultar Arqueo Con Parámetro Incorrecto



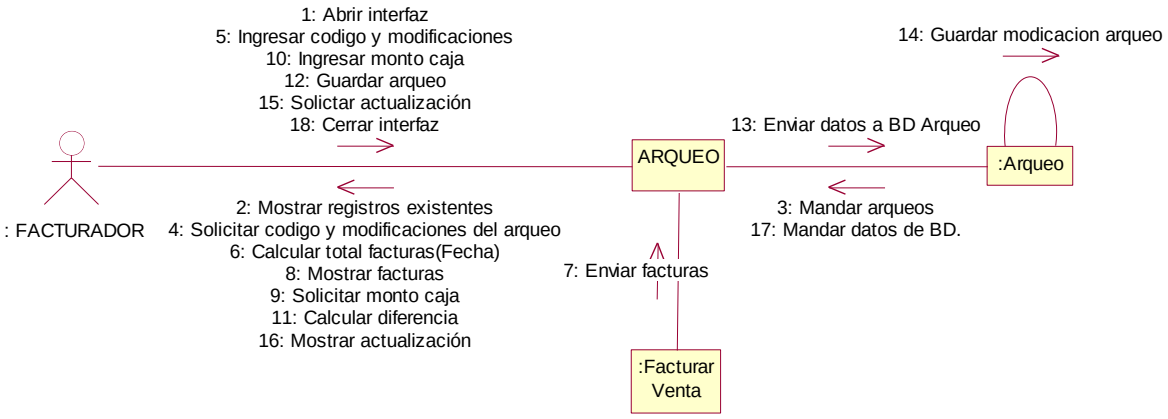

1.4. Diagrama de Colaboración: Consultar Arqueo Con Parametro Incorrecto



## Escenario 5: Modificar Arqueo

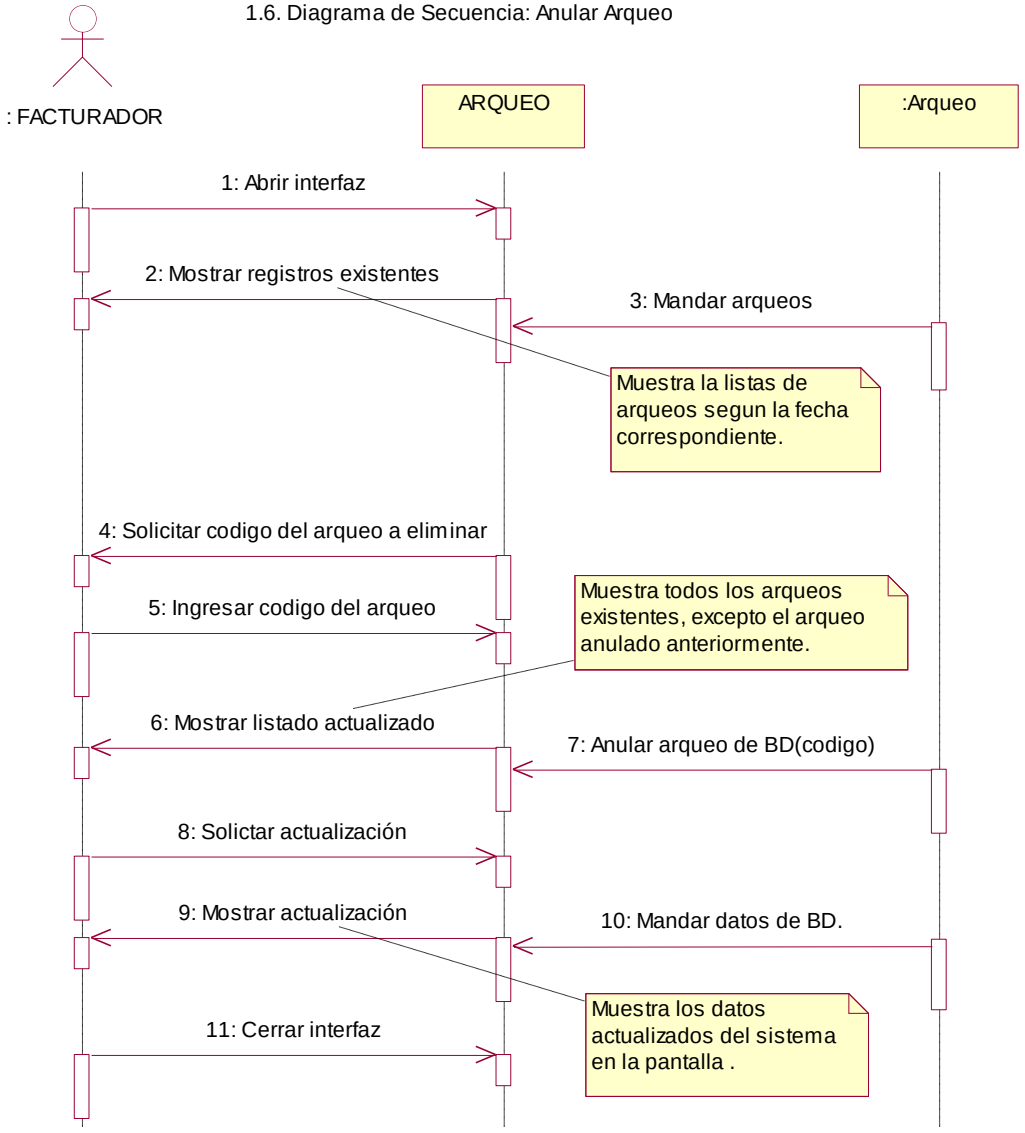


1.5. Diagrama de Colaboración: Modificar Arqueo

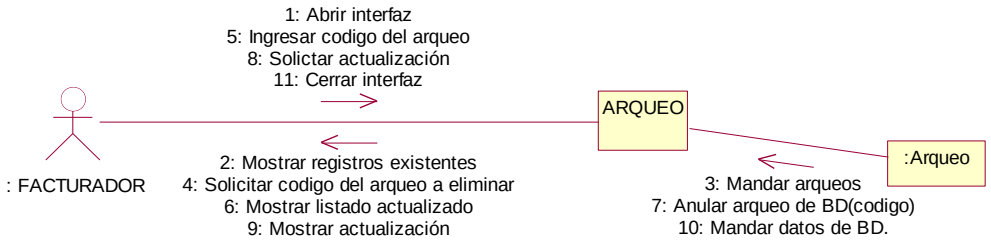


## Escenario 6: Anular Arqueo

1.6. Diagrama de Secuencia: Anular Arqueo



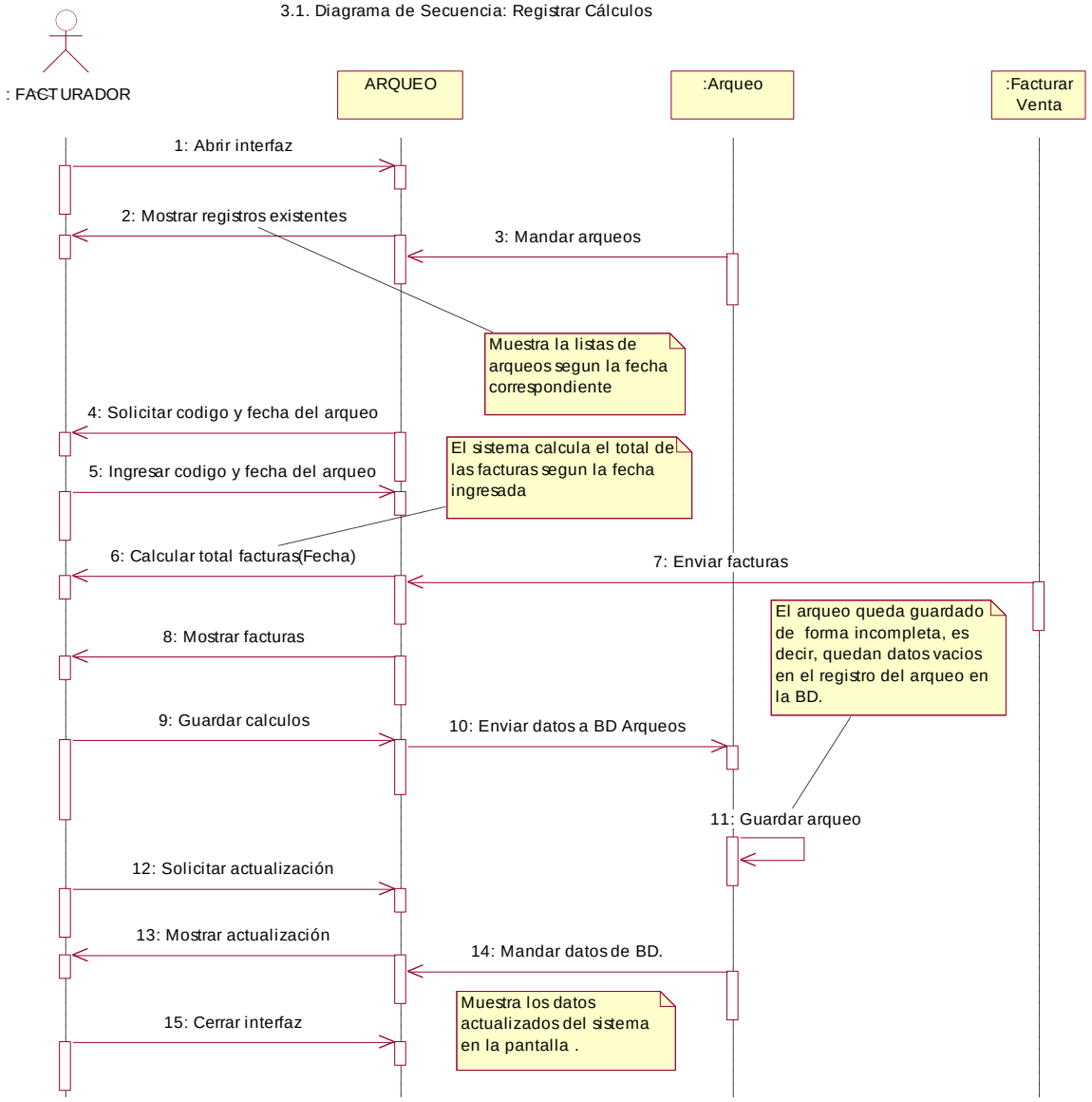

1.6. Diagrama de Colaboración: Anular Arqueo



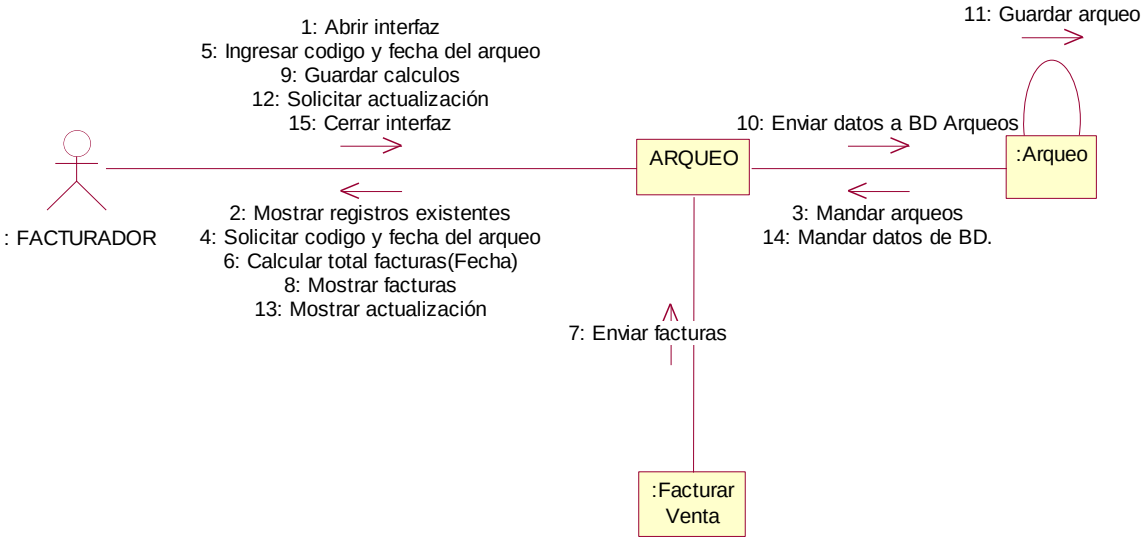
## Caso de Uso 3: Gestionar Monto Total Factura por Día

## Escenario 1: Registrar Cálculos

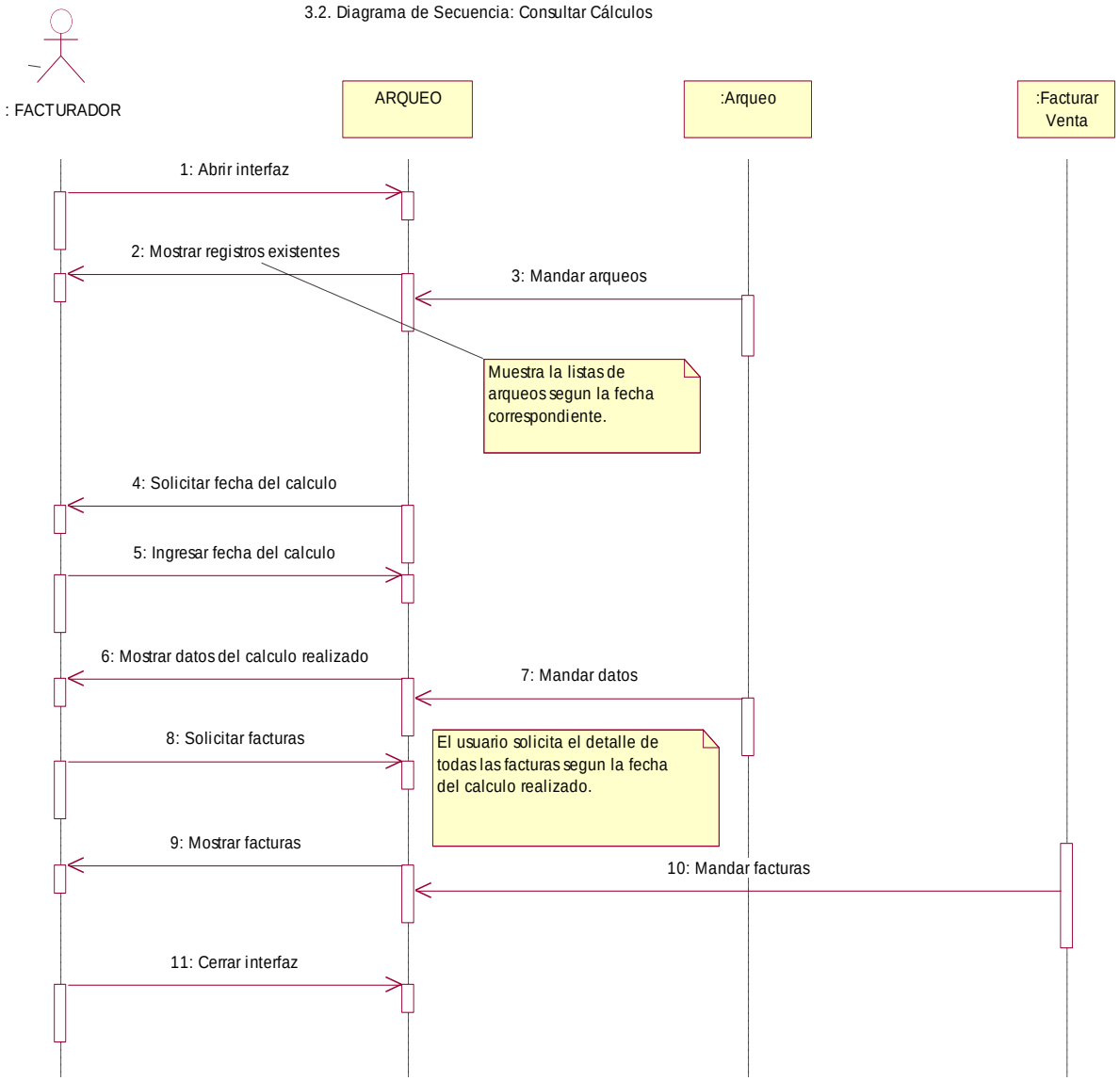
3.1. Diagrama de Secuencia: Registrar Cálculos



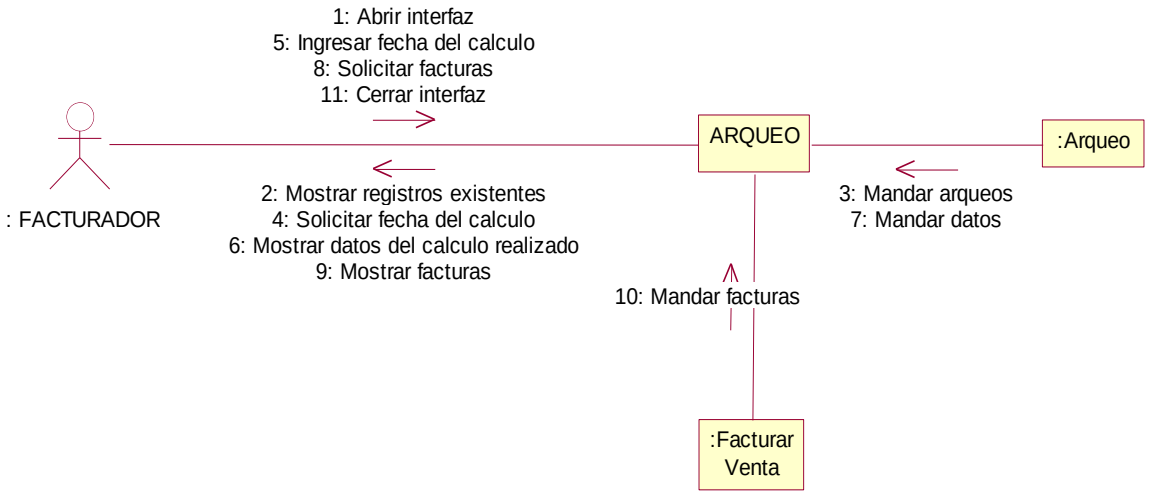
3.1. Diagrama de Colaboración: Registrar Cálculos



## Escenario 2: Consultar Cálculo



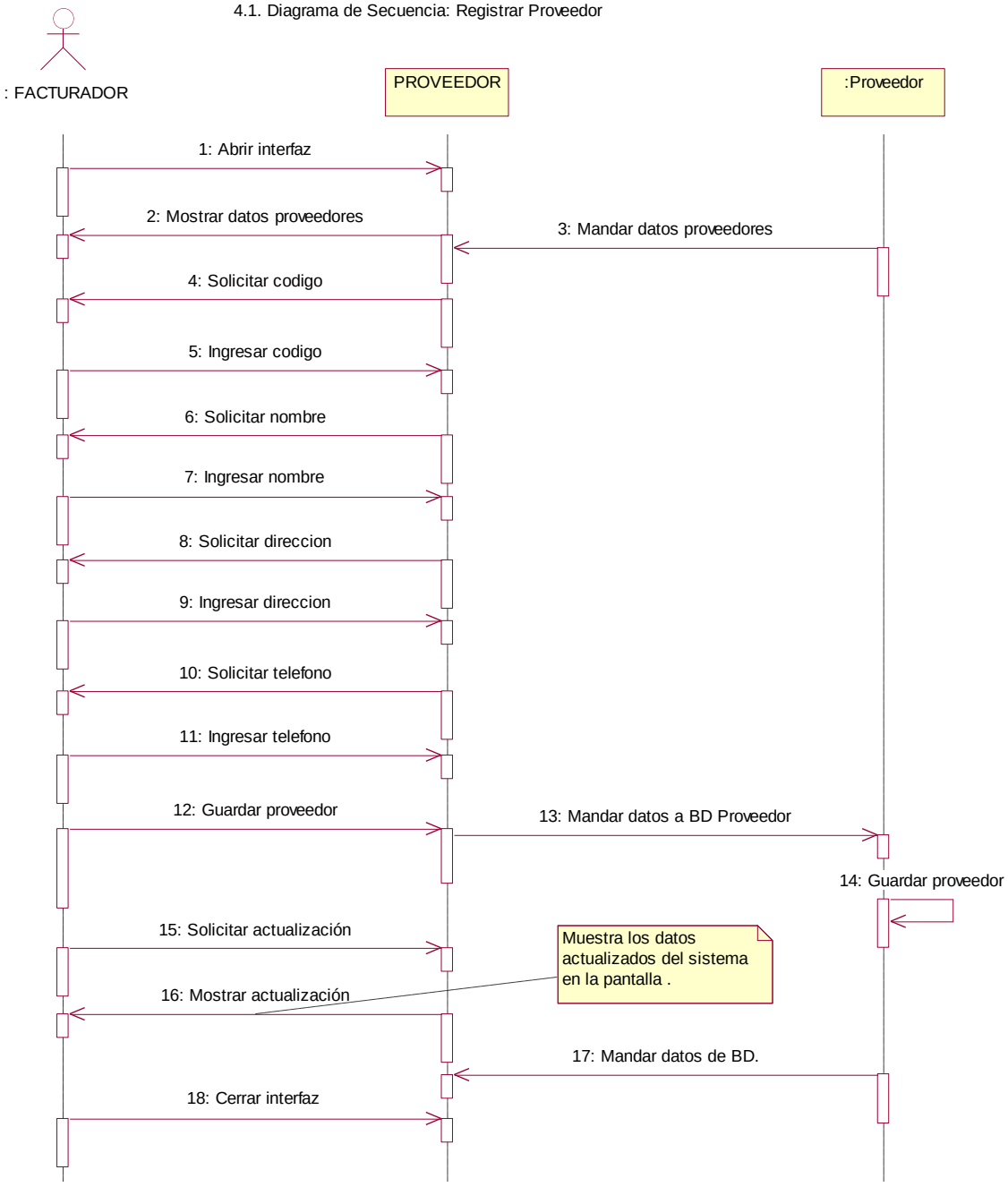
3.2. Diagrama de Colaboración: Consultar Cálculos



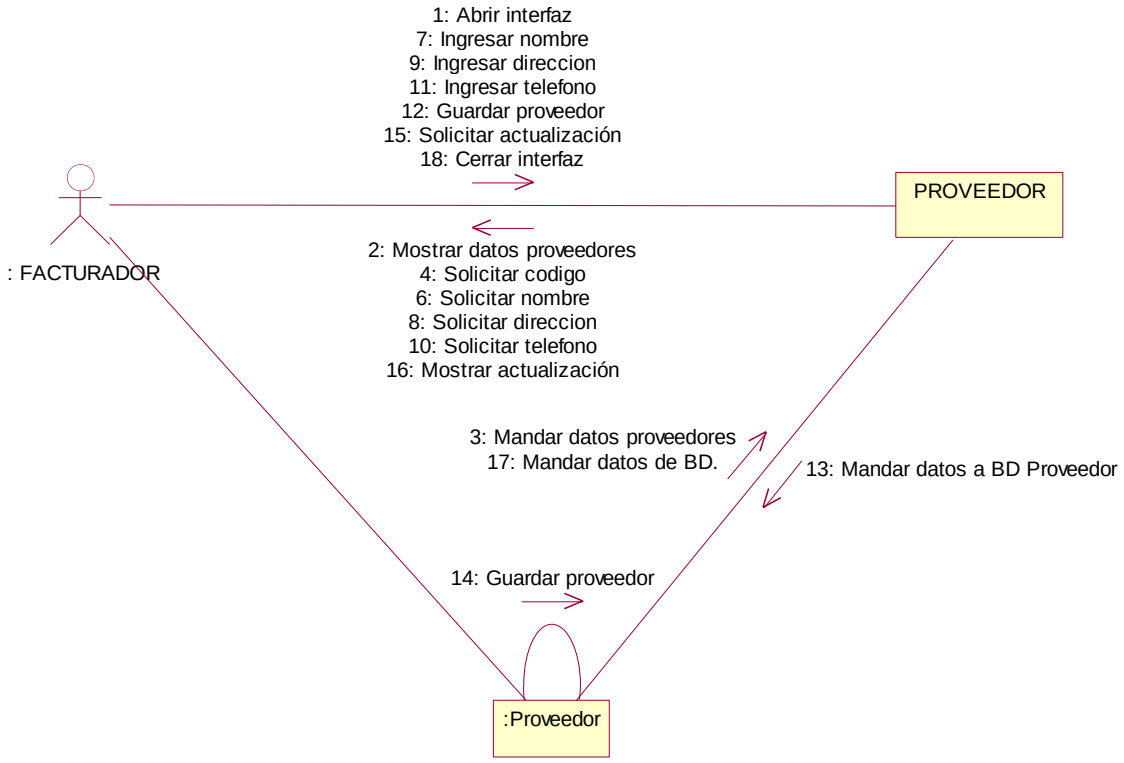
## Caso de Uso 4: Gestionar Proveedores

## Escenario 1: Registrar Proveedor

4.1. Diagrama de Secuencia: Registrar Proveedor

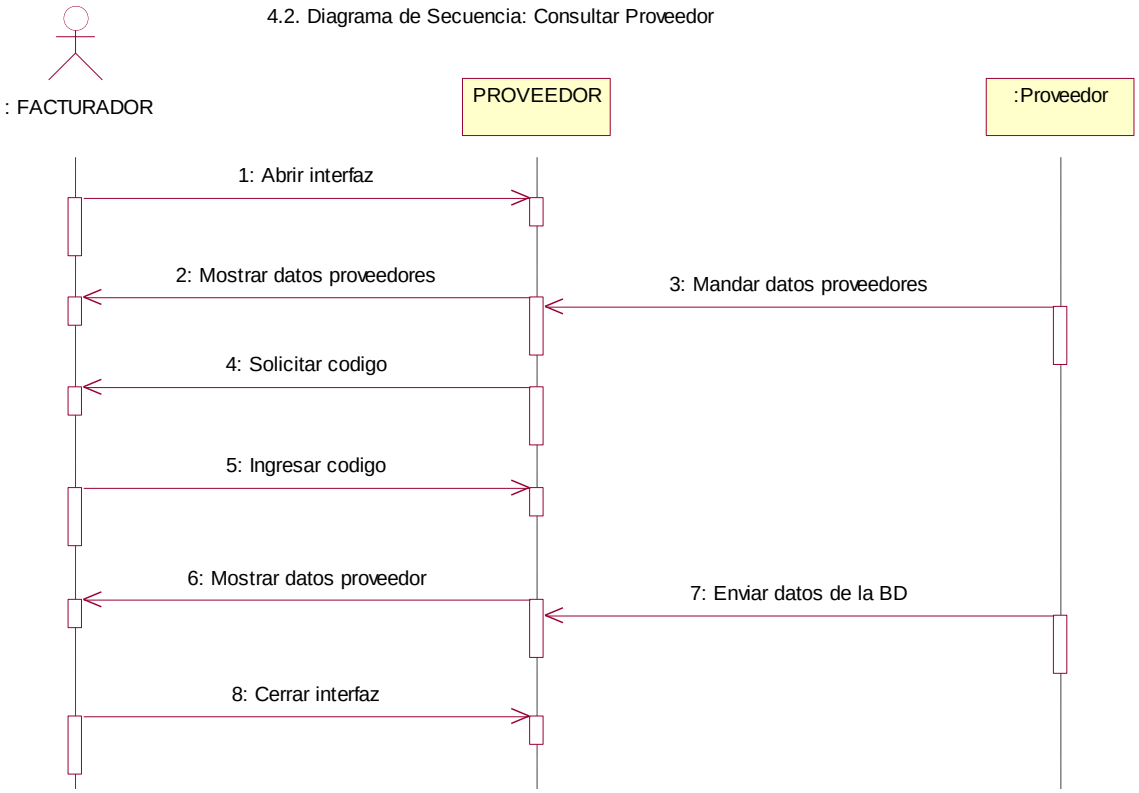


4.1. Diagrama de Colaboración: Registrar Proveedor

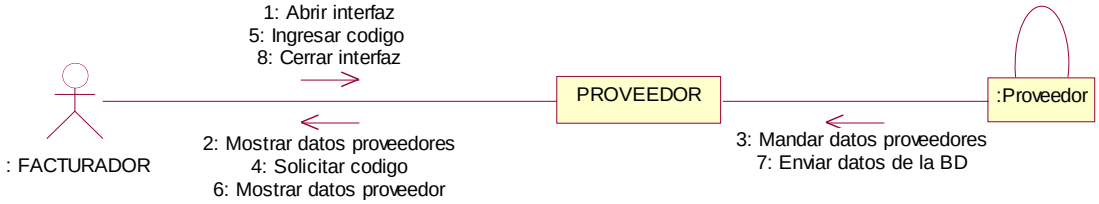


## Escenario 2: Consultar Proveedor

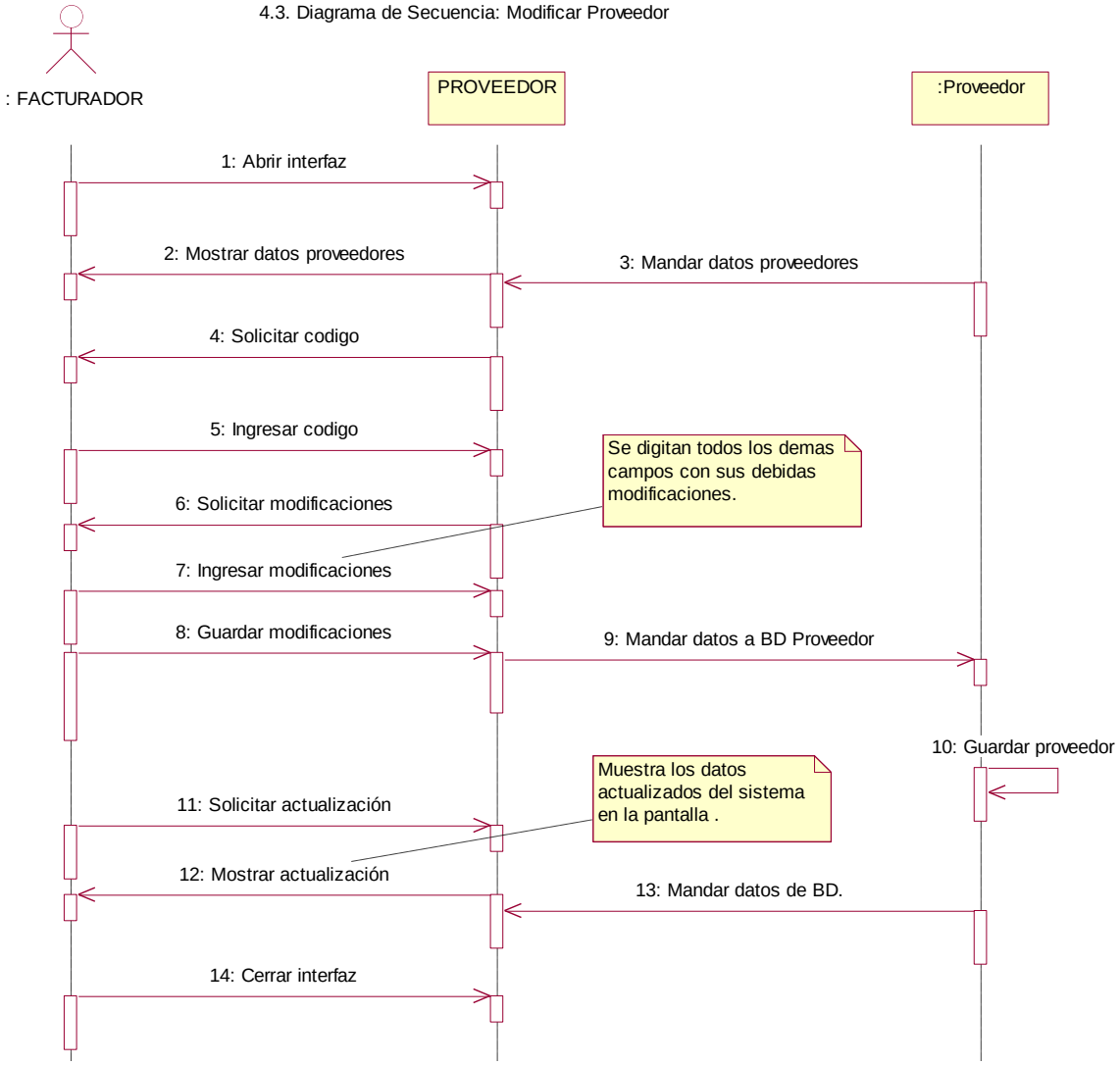
4.2. Diagrama de Secuencia: Consultar Proveedor



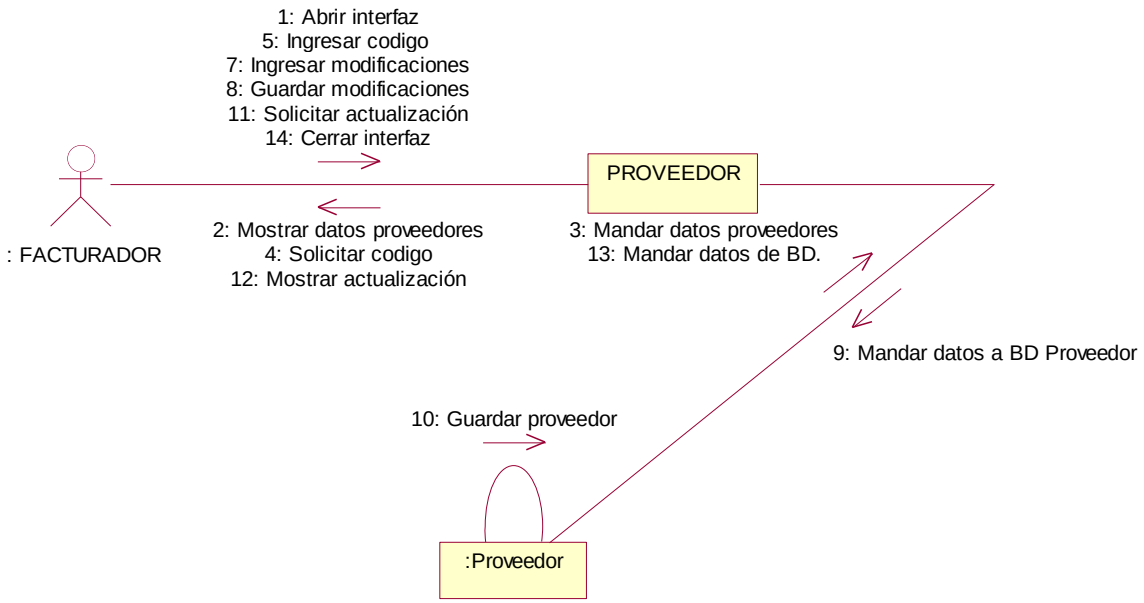

4.2. Diagrama de Colaboración: Consultar Proveedor



## Escenario 3: Modificar Proveedor

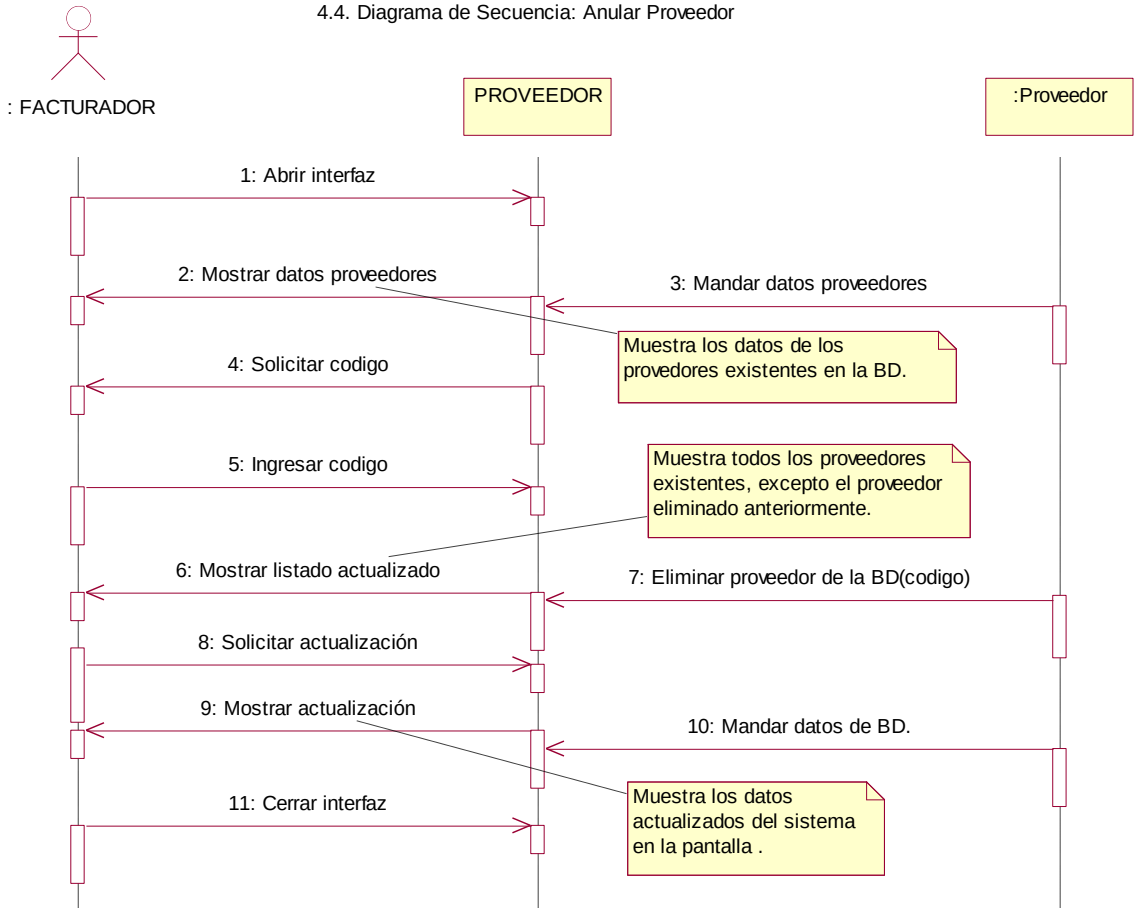


4.3. Diagrama de Colaboración: Modificar Proveedor

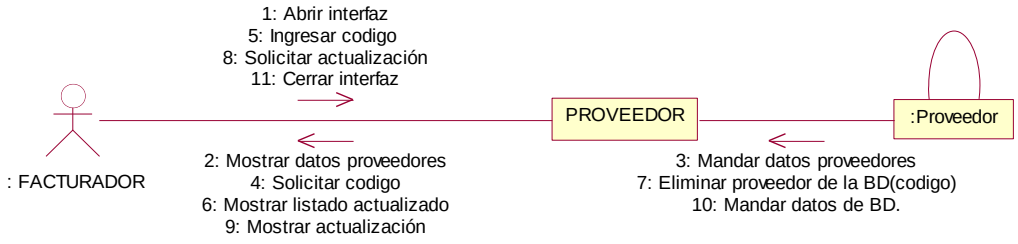


## Escenario 4: Eliminar Proveedor

4.4. Diagrama de Secuencia: Anular Proveedor



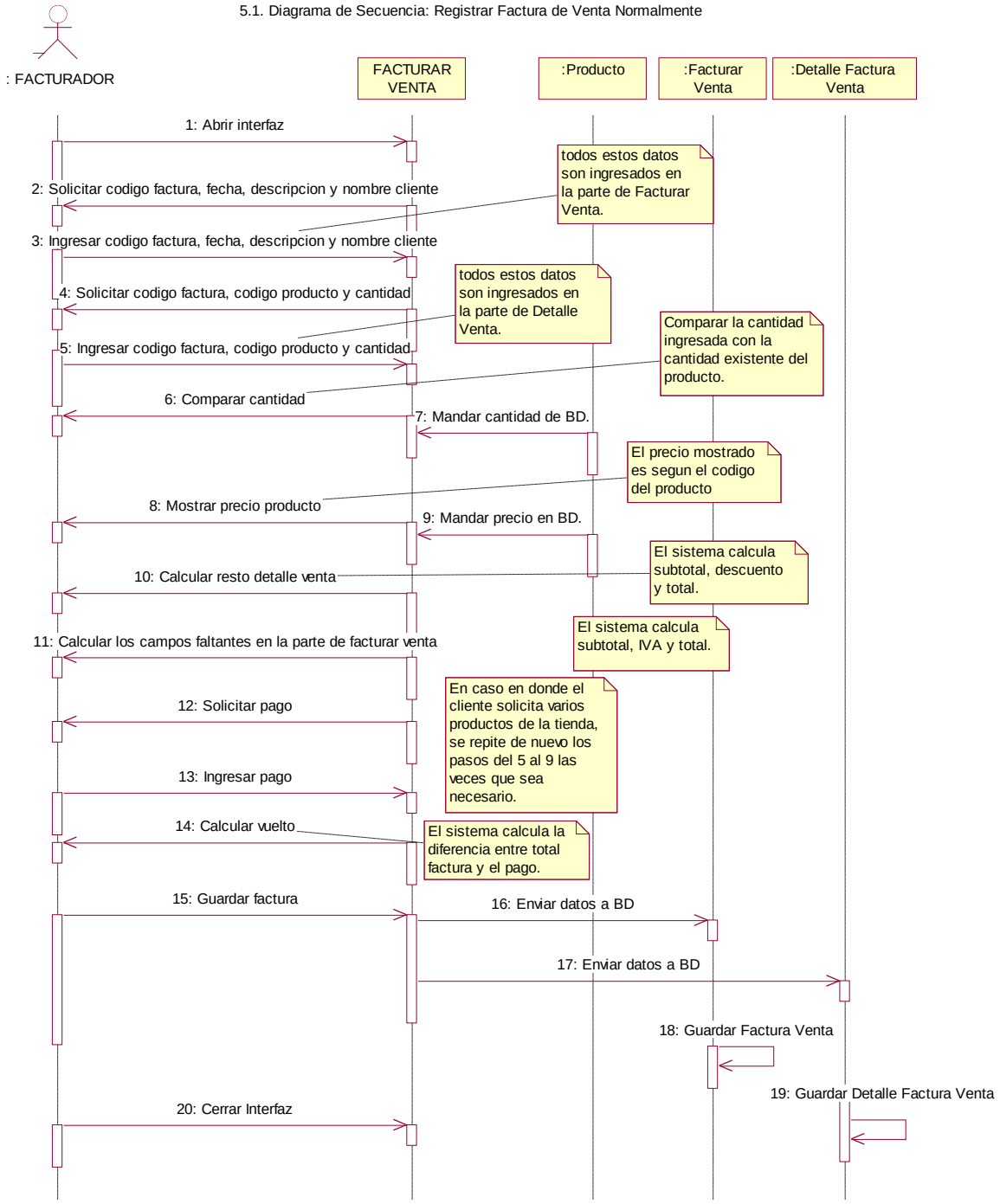

4.4. Diagrama de Colaboración: Anular Proveedor



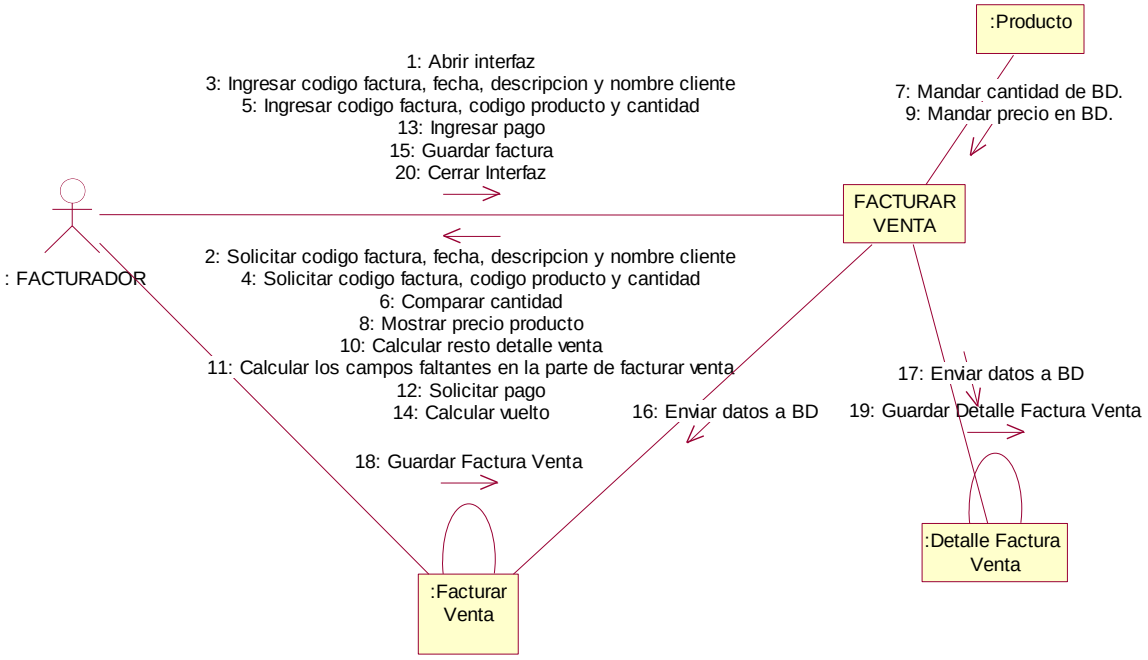
## Caso de Uso 5: Gestionar Factura Venta

## Escenario 1: Registrar Factura de Venta Normalmente

5.1. Diagrama de Secuencia: Registrar Factura de Venta Normalmente

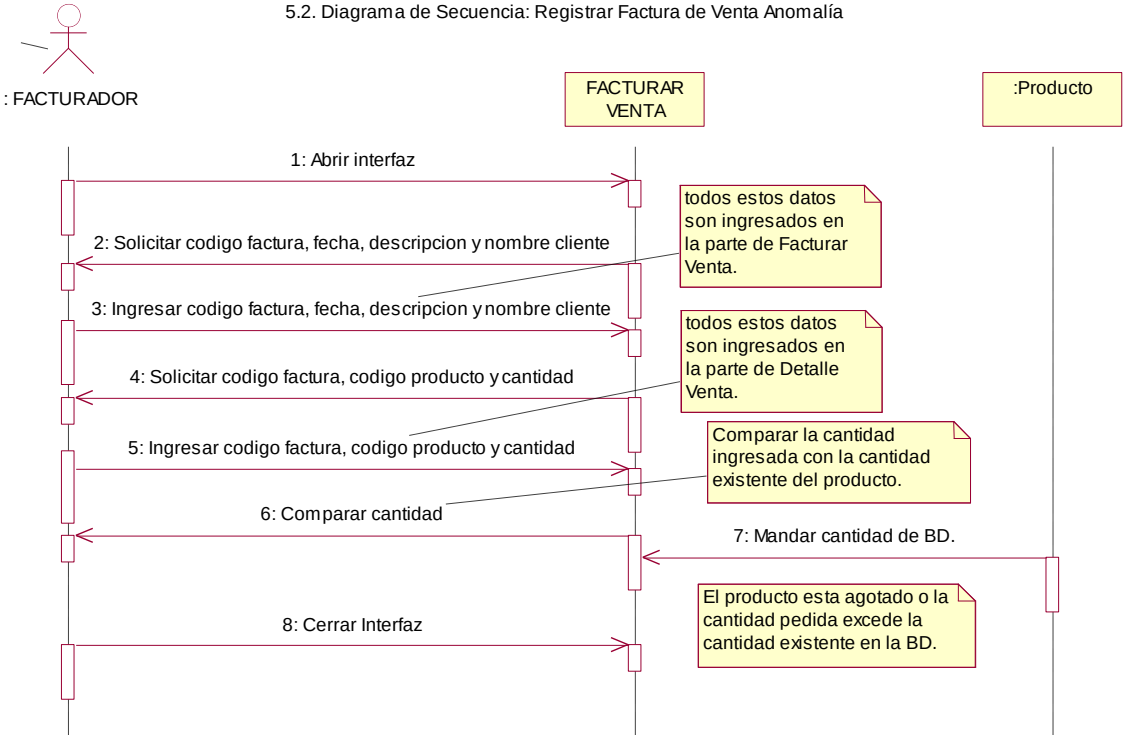


5.1. Diagrama de Colaboración: Registrar Factura de Venta Normalmente

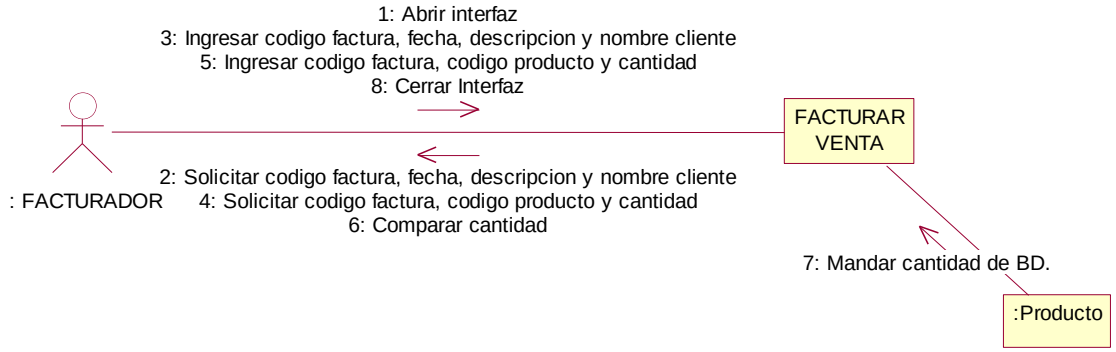


## Escenario 2: Registrar Factura de Venta Anomalía

5.2. Diagrama de Secuencia: Registrar Factura de Venta Anomalía

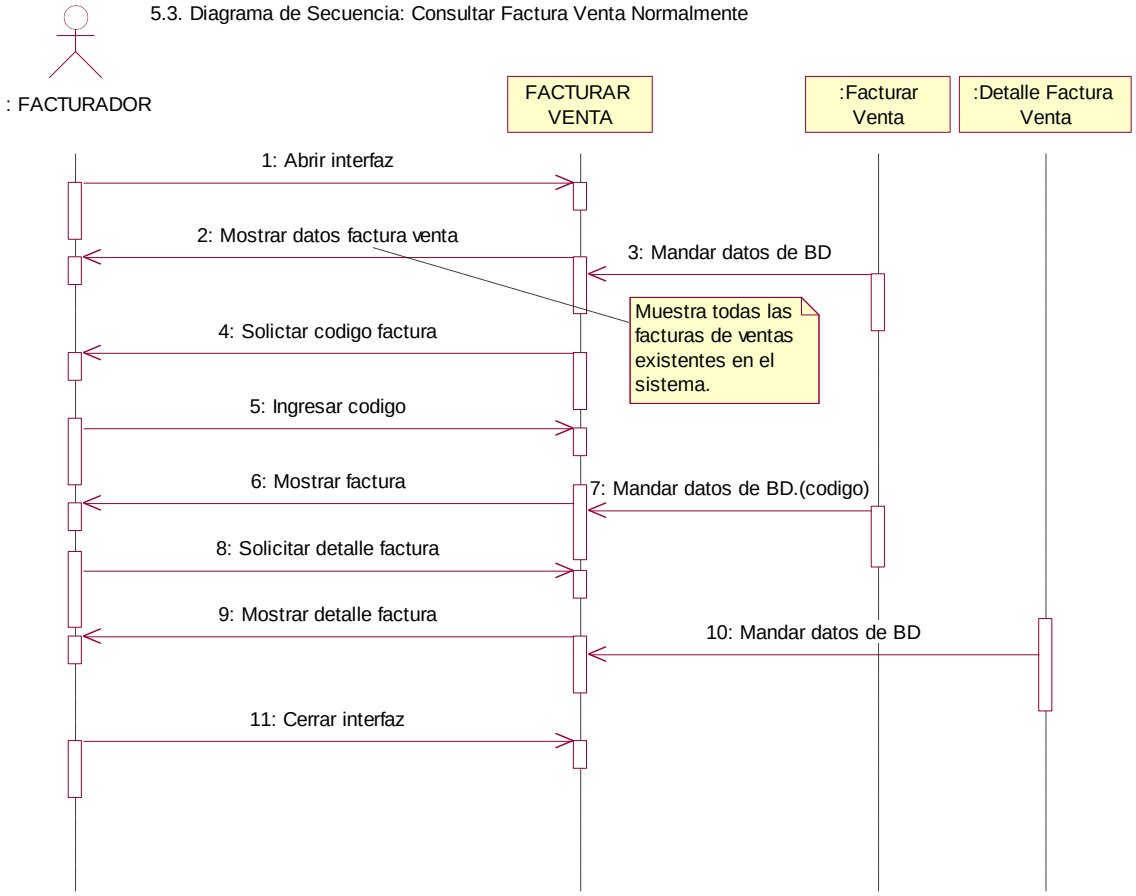



5.2. Diagrama de Colaboración: Registrar Factura de Venta Anomalía

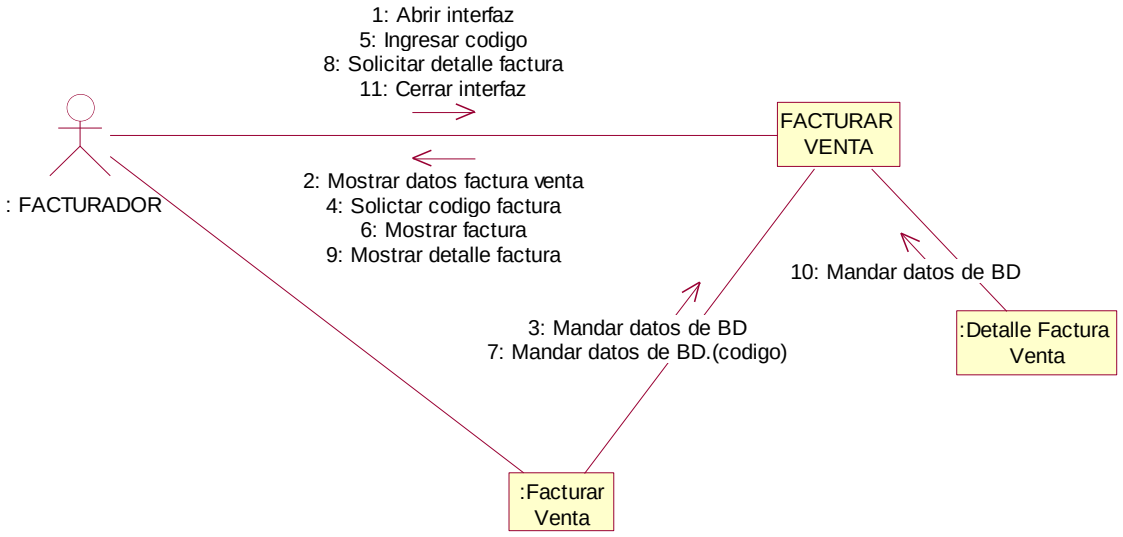


## Escenario 3: Consultar Factura Venta Normalmente

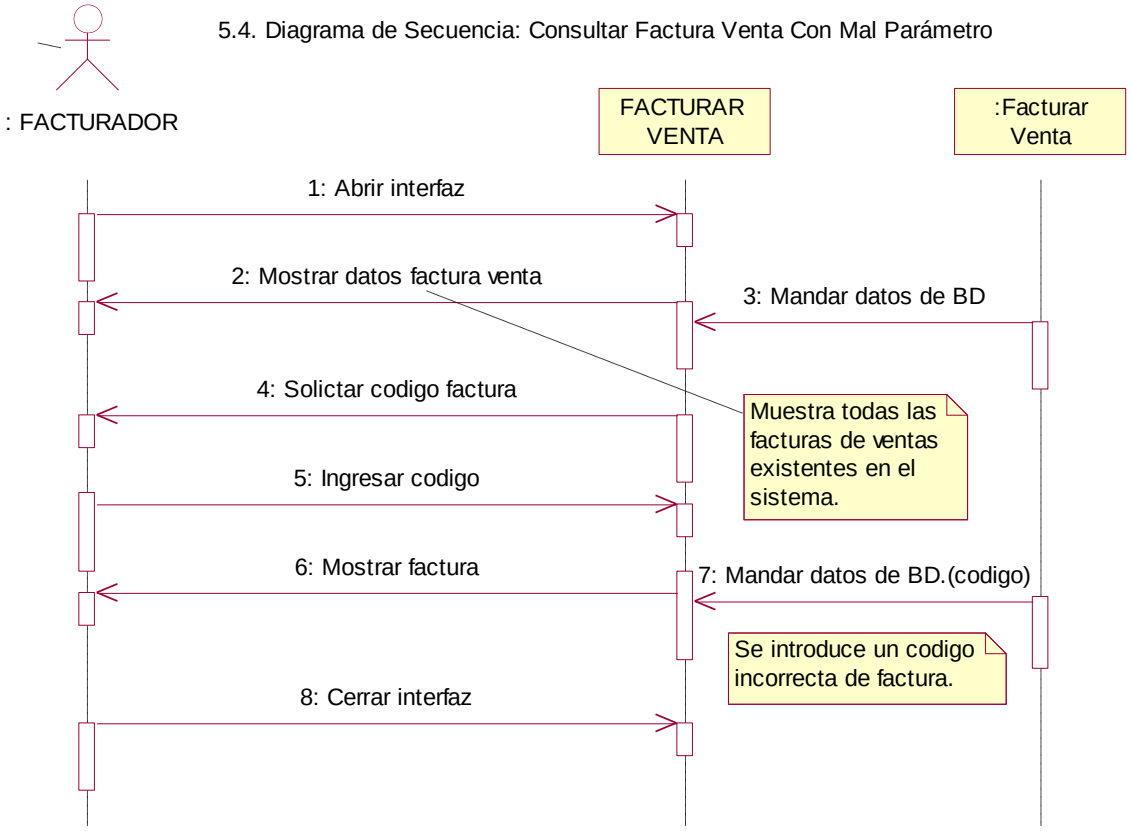
5.3. Diagrama de Secuencia: Consultar Factura Venta Normalmente



5.3. Diagrama de Colaboración: Consultar Factura Venta Normalmente

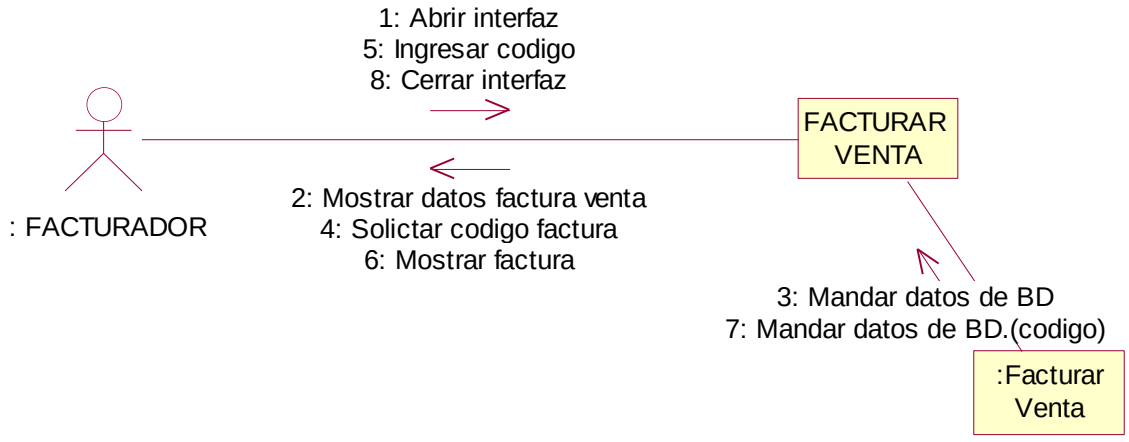


## Escenario 4: Consultar Factura Venta con Mal Parámetro



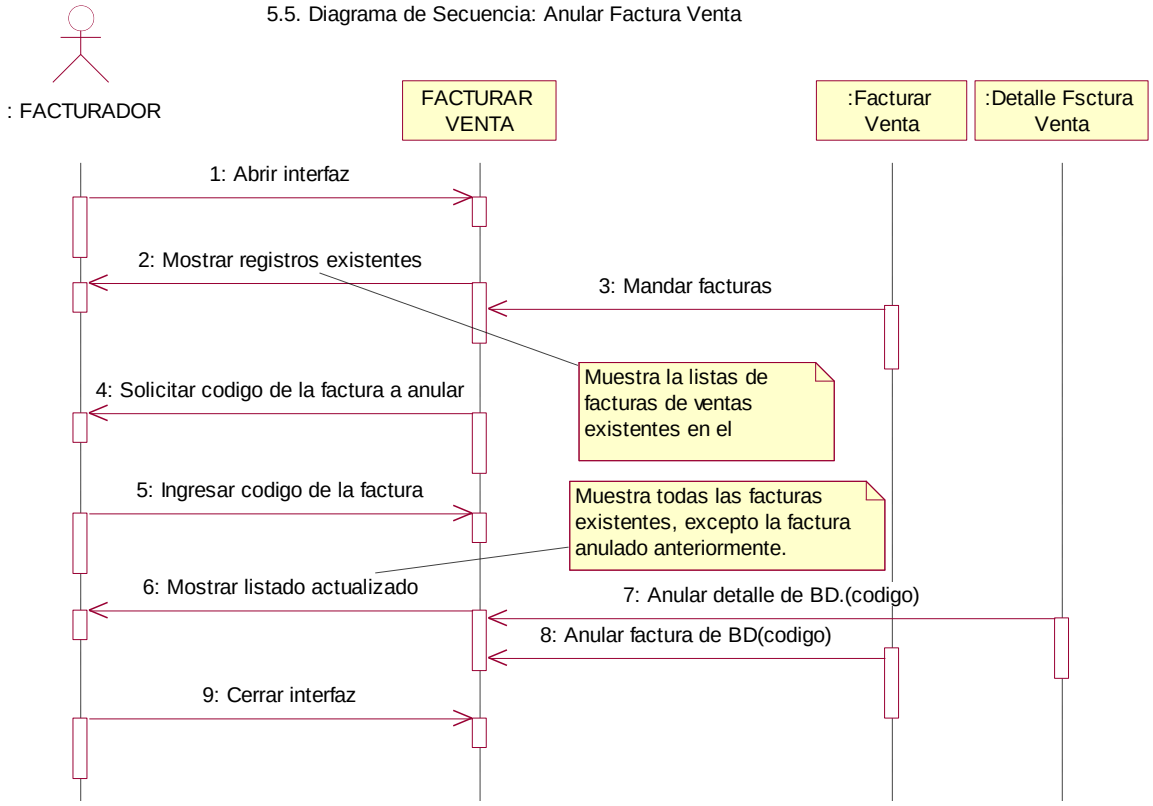

5.4. Diagrama de Secuencia: Consultar Factura Venta Con Mal Parámetro

5.4. Diagrama de Colaboración: Consultar Factura Venta Con Mal Parámetro

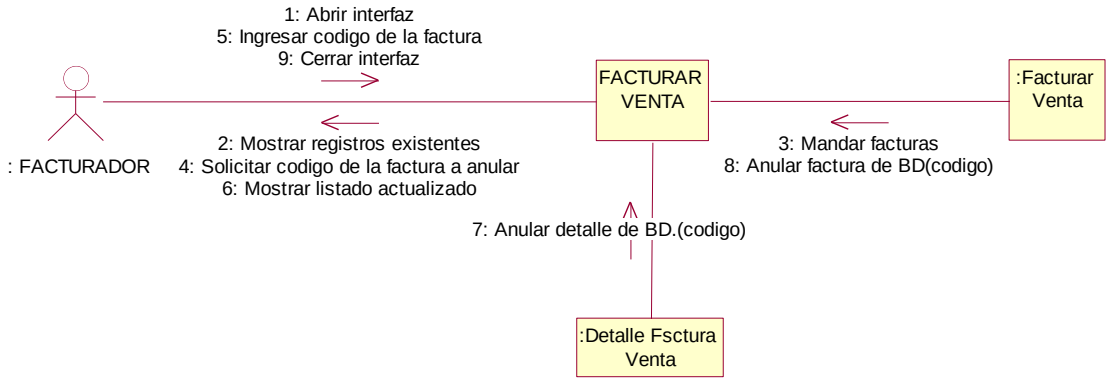


## Escenario 5: Anular Factura Venta

5.5. Diagrama de Secuencia: Anular Factura Venta



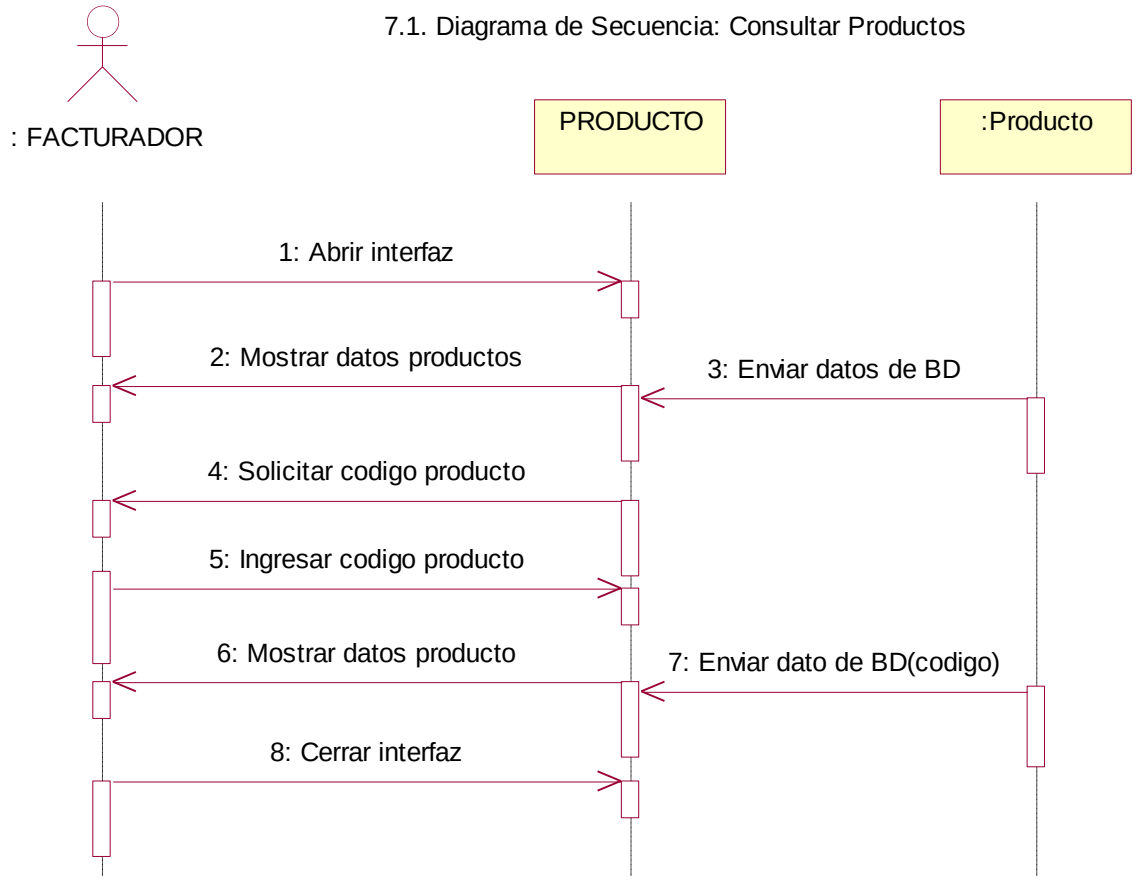

5.5. Diagrama de Colaboración: Anular Factura Venta



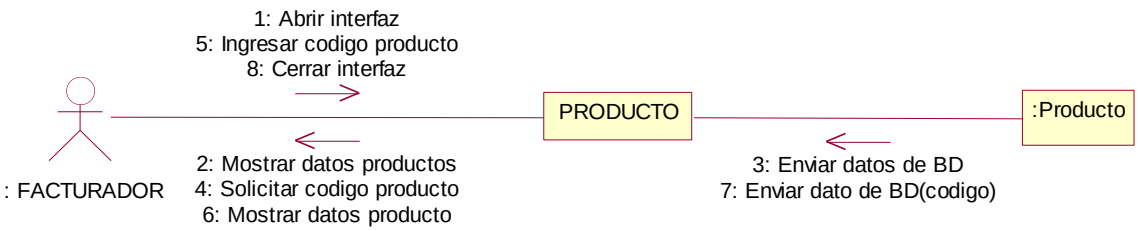
## Caso de Uso 7: Gestionar Catalogo Productos

## Escenario 1: Consultar Productos

7.1. Diagrama de Secuencia: Consultar Productos

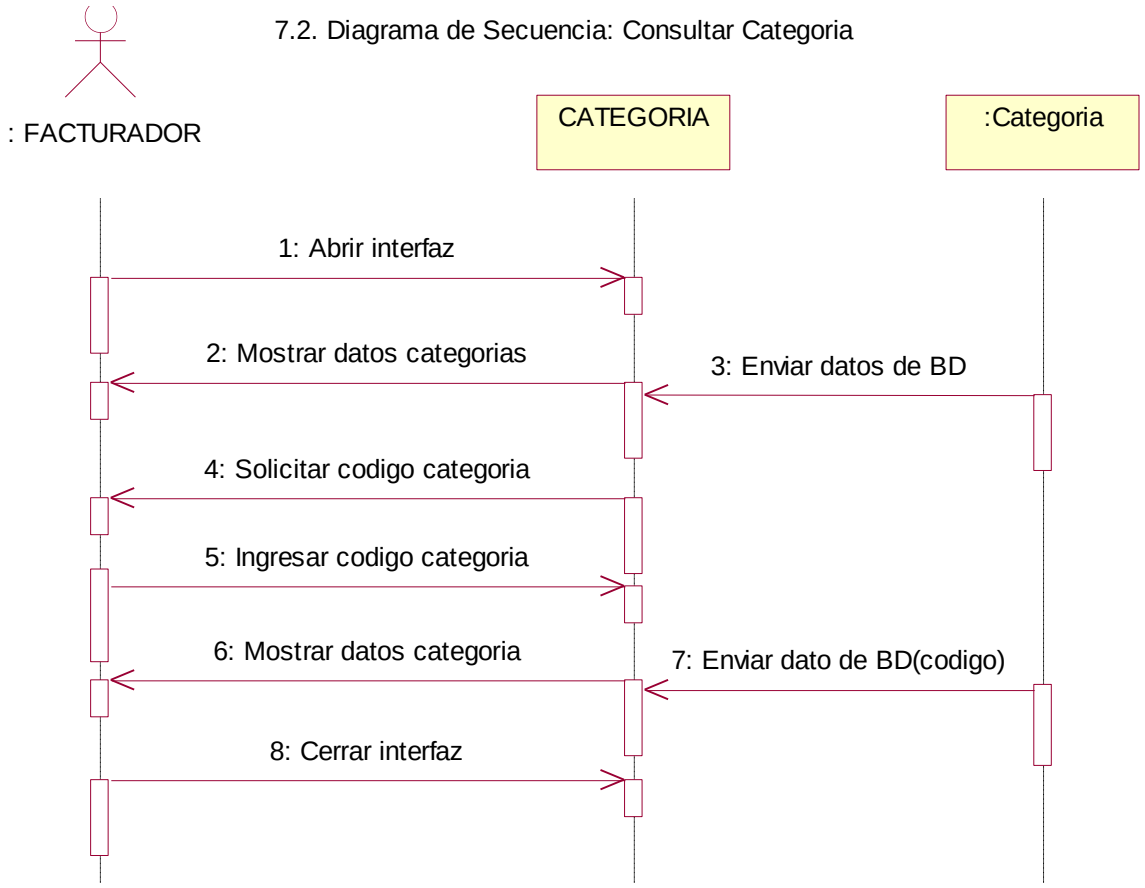



7.1. Diagrama de Colaboración: Consultar Catalogo Productos

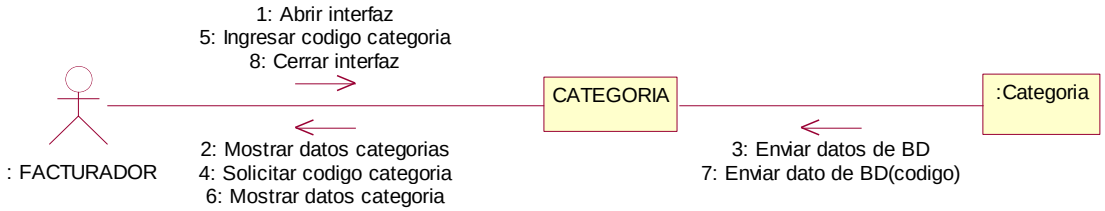


## Escenario 2: Consultar Categoría

7.2. Diagrama de Secuencia: Consultar Categoria

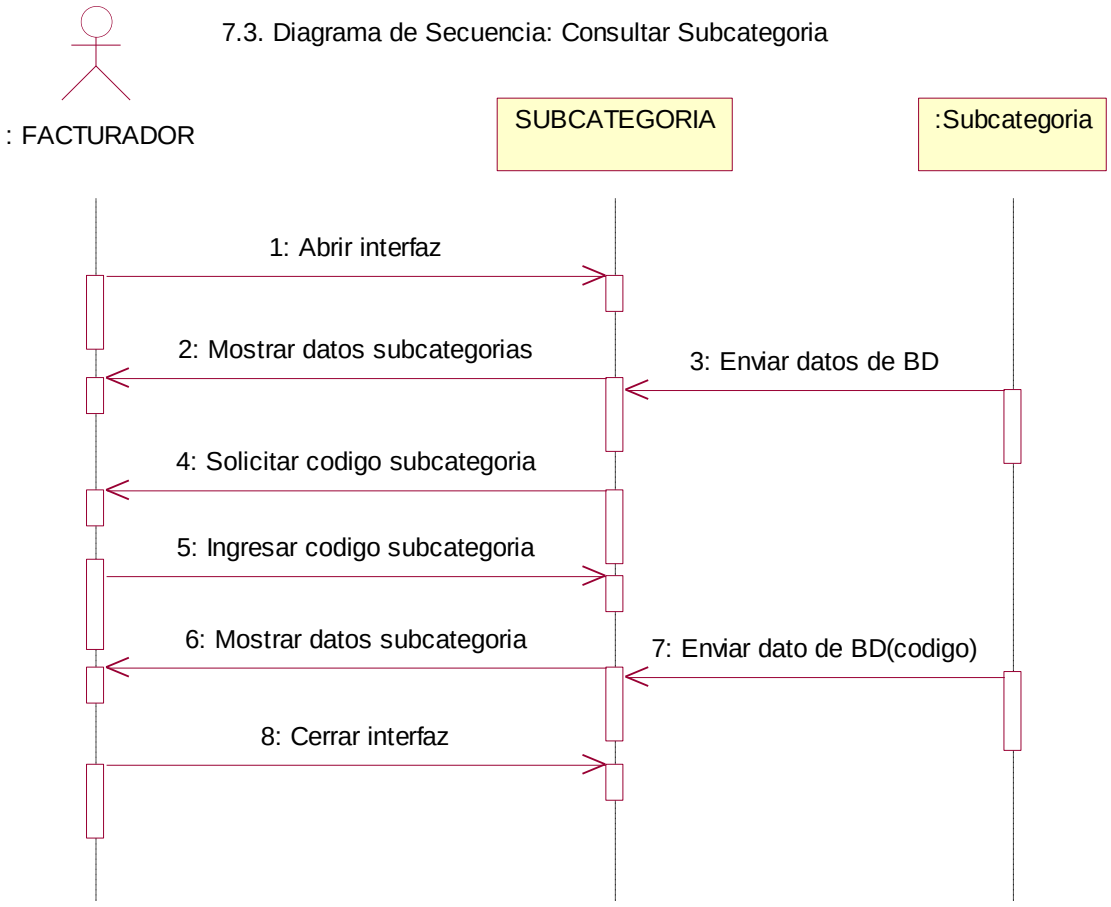



7.2. Diagrama de Colaboración: Consultar Categoria

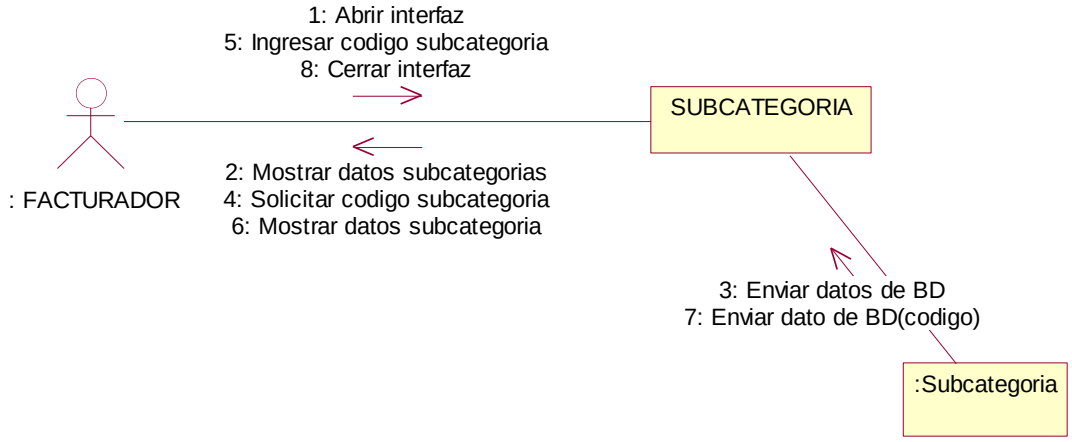


## Escenario 3: Consultar Subcategoría

7.3. Diagrama de Secuencia: Consultar Subcategoria



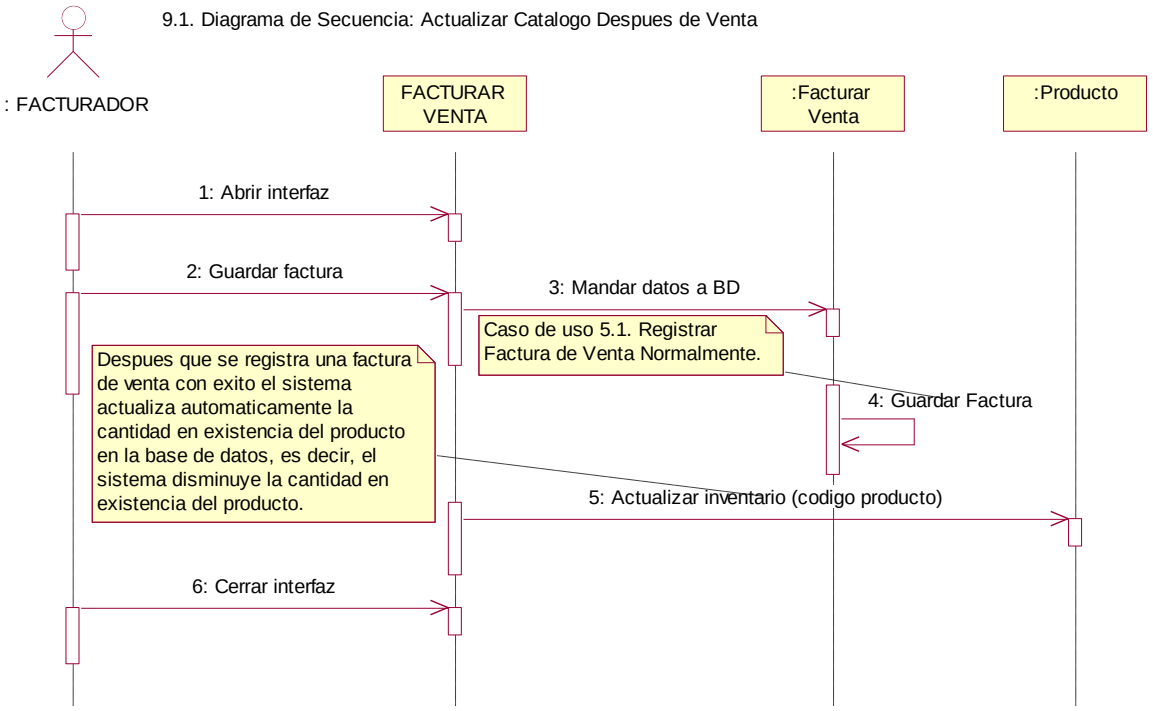

7.3. Diagrama de Colaboración: Consultar Subcategoria



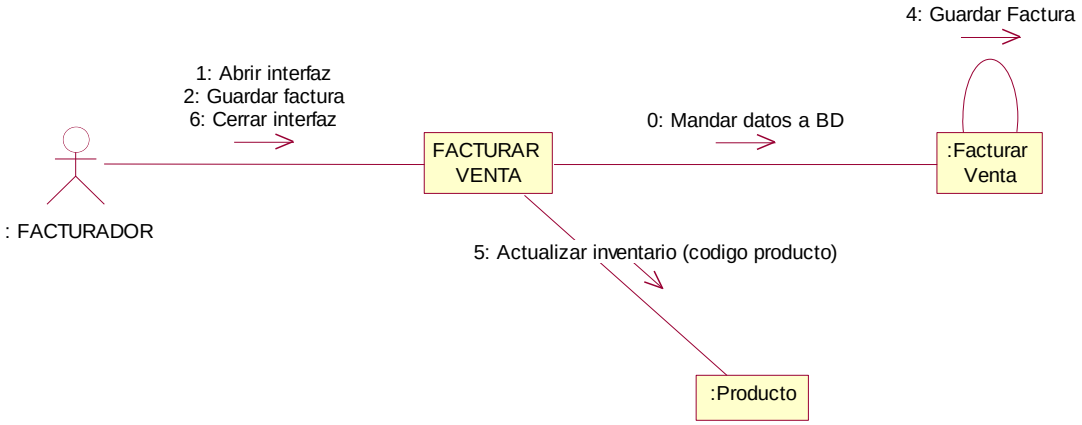
## Caso de Uso 9: Gestionar Movimientos Productos

## Escenario 1: Actualizar Catalogo Después de Venta

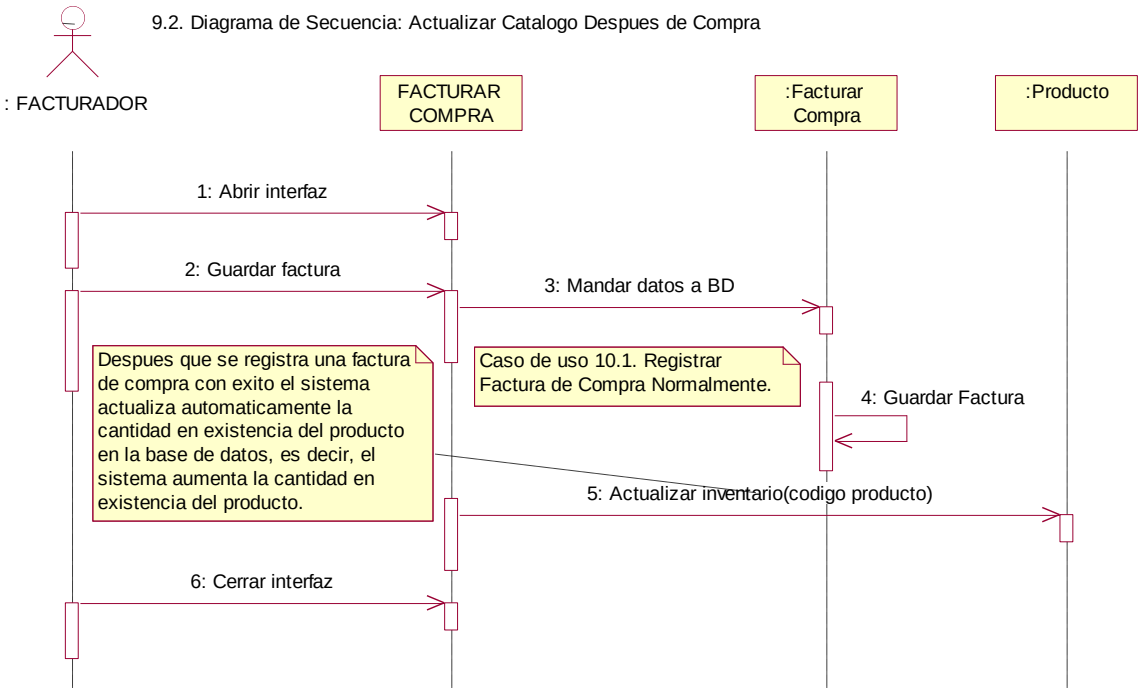
9.1. Diagrama de Secuencia: Actualizar Catalogo Despues de Venta



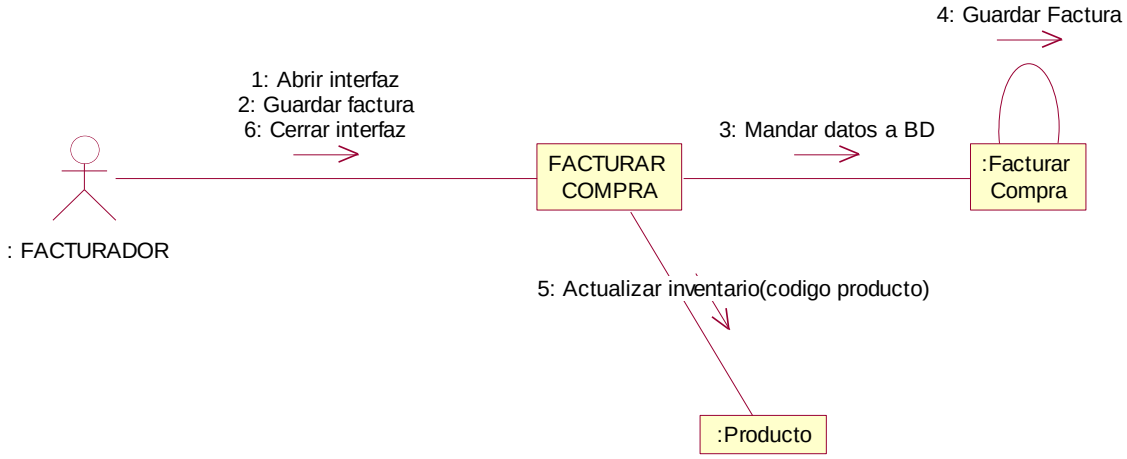

9.1. Diagrama de Colaboración: Actualizar Catalogo Despues de Venta



## Escenario 2: Actualizar Catalogo Después de Comprar

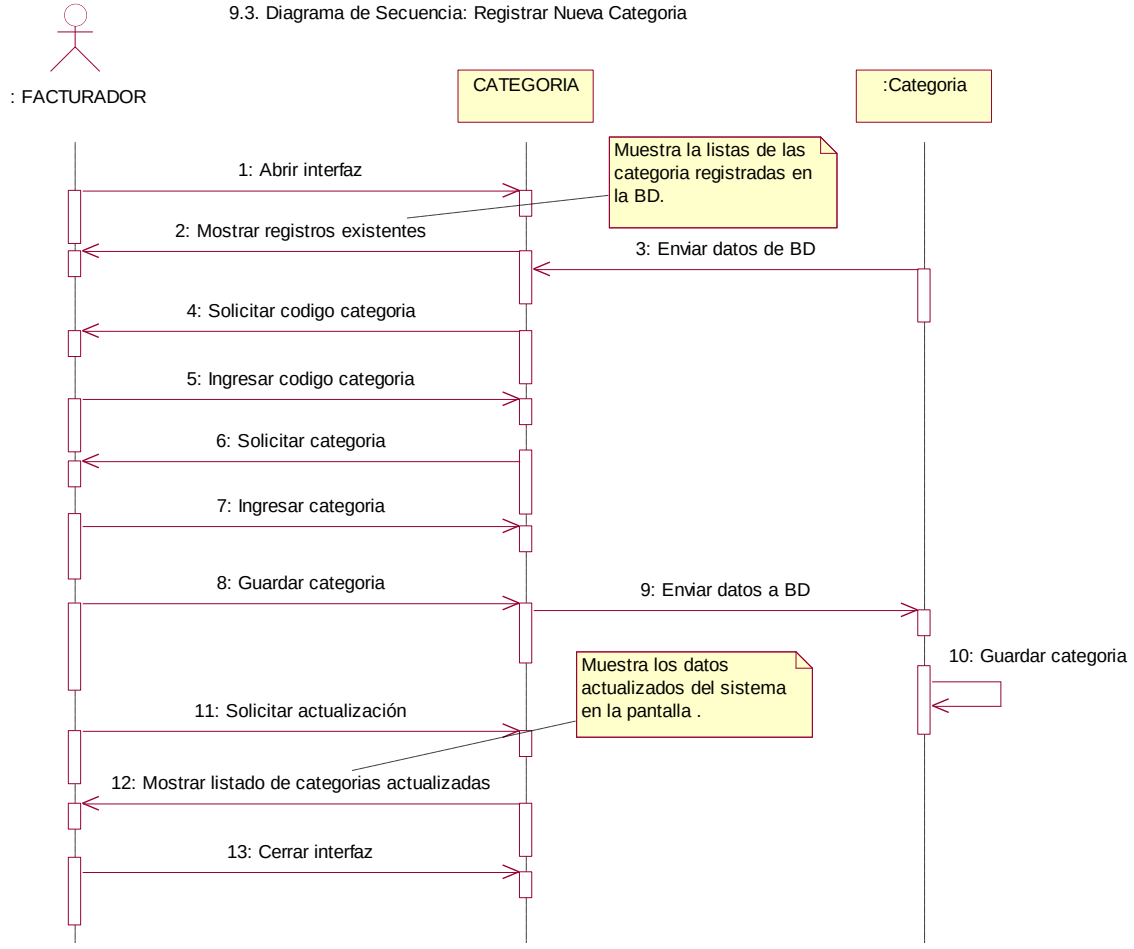



9.2. Diagrama de Colaboración: Actualizar Catalogo Despues de Compra

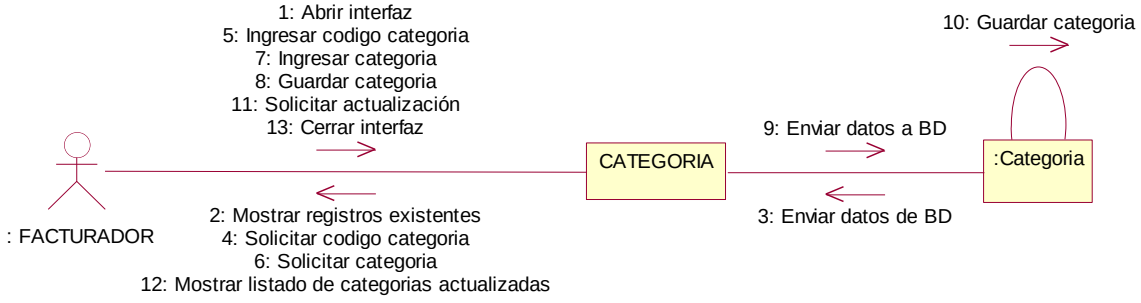


## Escenario 3: Registrar nueva categoría

9.3. Diagrama de Secuencia: Registrar Nueva Categoria

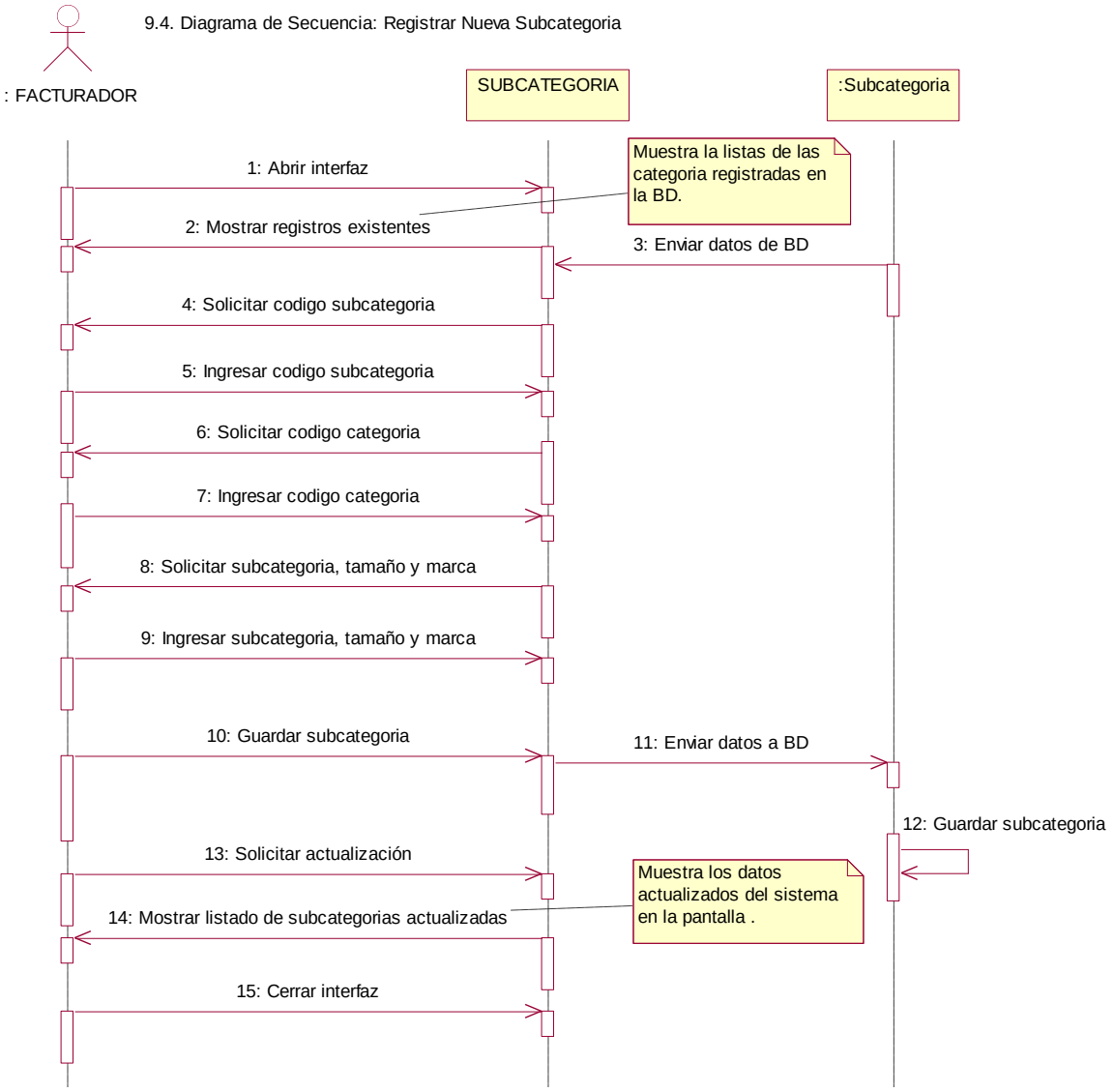



9.3. Diagrama de Colaboración: Registrar Nueva Categoria

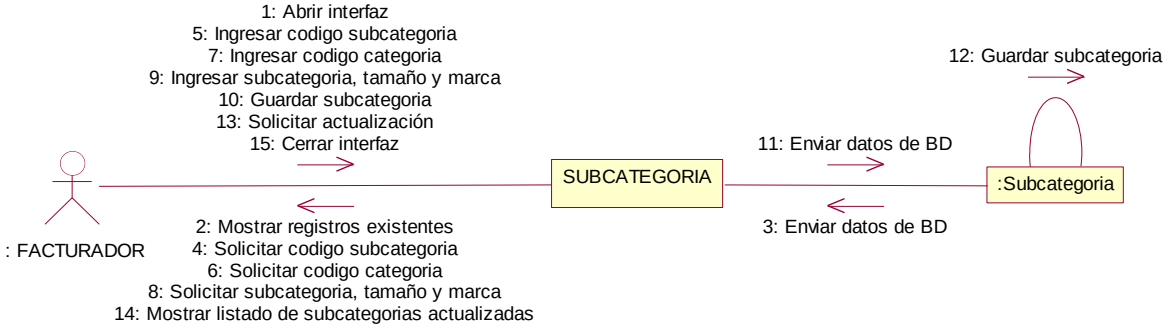


## Escenario 4: Registrar nueva subcategoría

9.4. Diagrama de Secuencia: Registrar Nueva Subcategoria

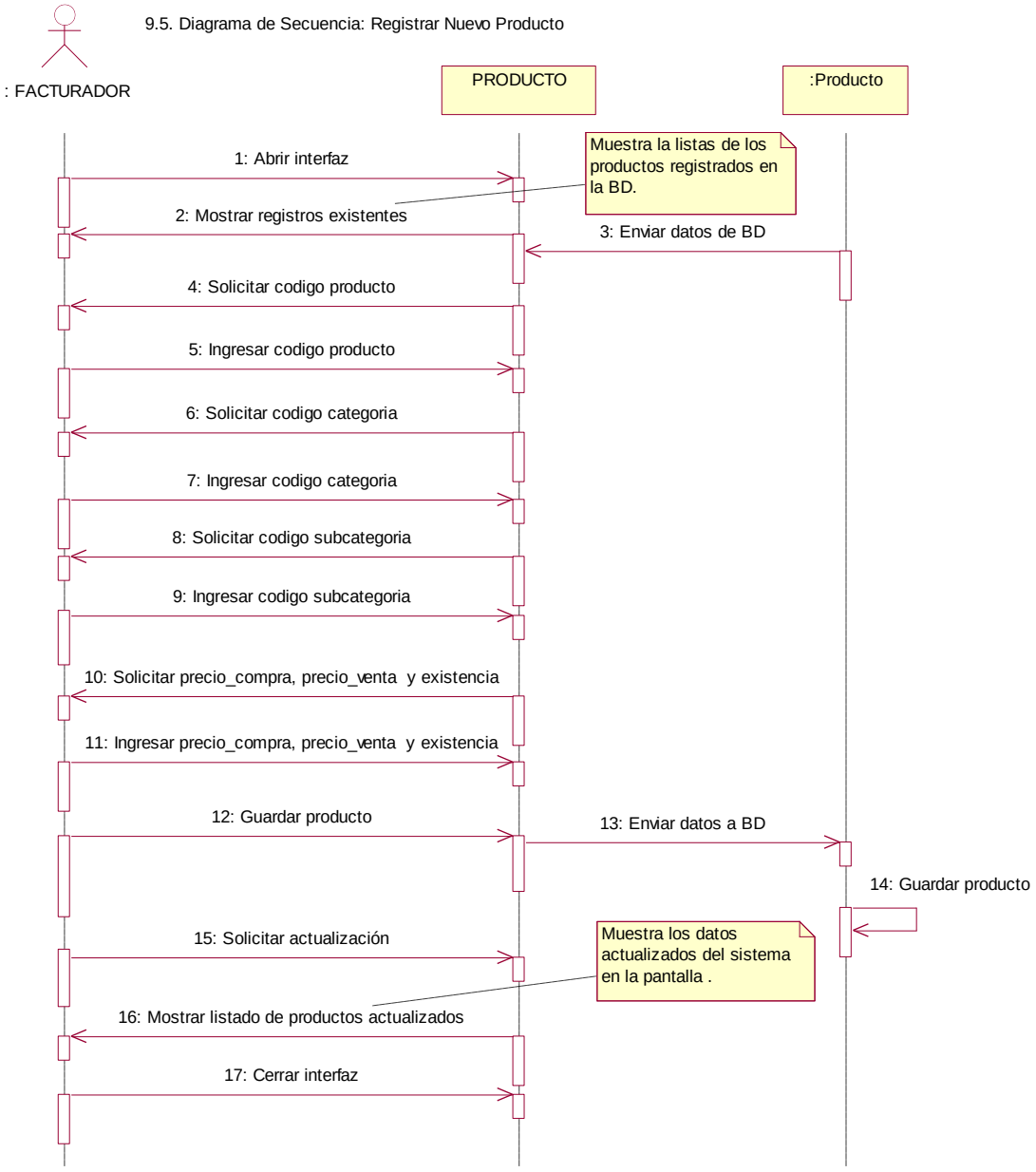



9.4. Diagrama de Colaboración: Registrar Nueva Subcategoria

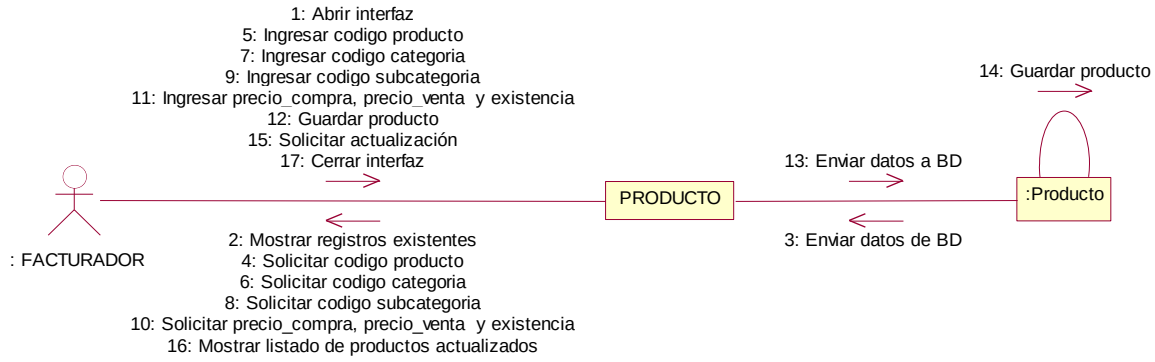


## Escenario 5: Registrar nuevo producto

9.5. Diagrama de Secuencia: Registrar Nuevo Producto

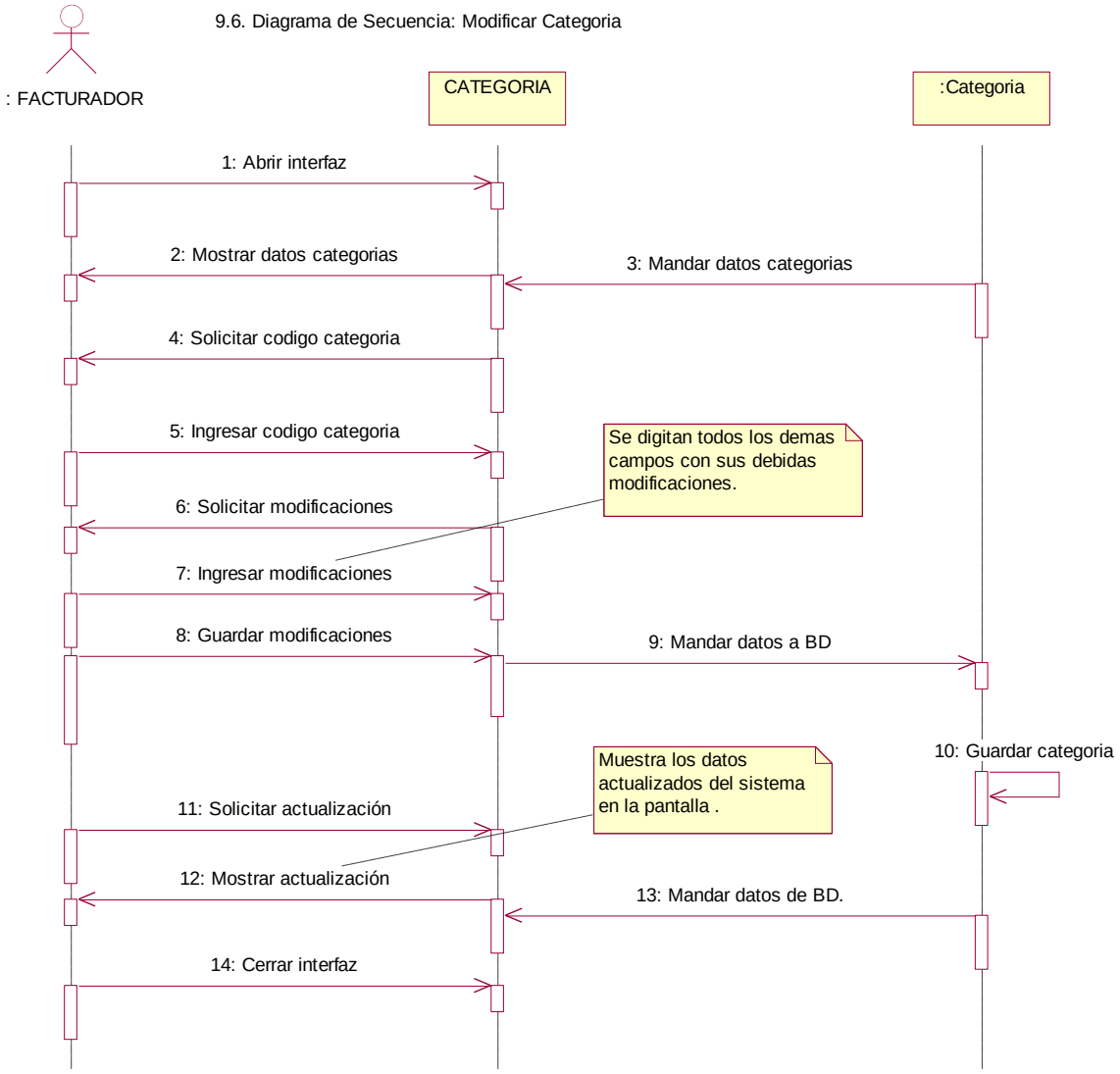



9.5. Diagrama de Colaboración: Registrar Nuevo Producto

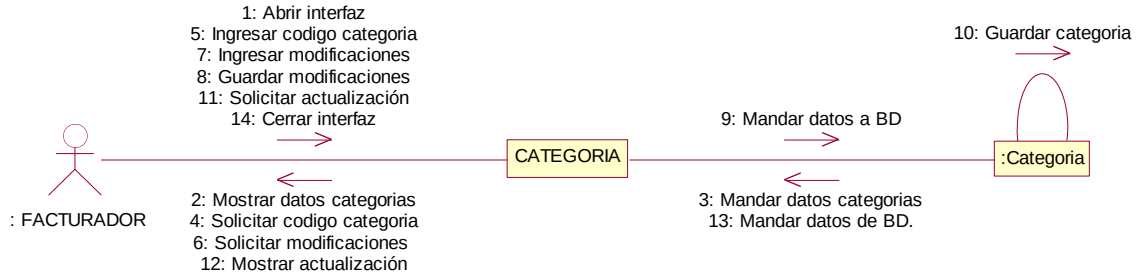


## Escenario 6: Modificar Categoría

9.6. Diagrama de Secuencia: Modificar Categoria

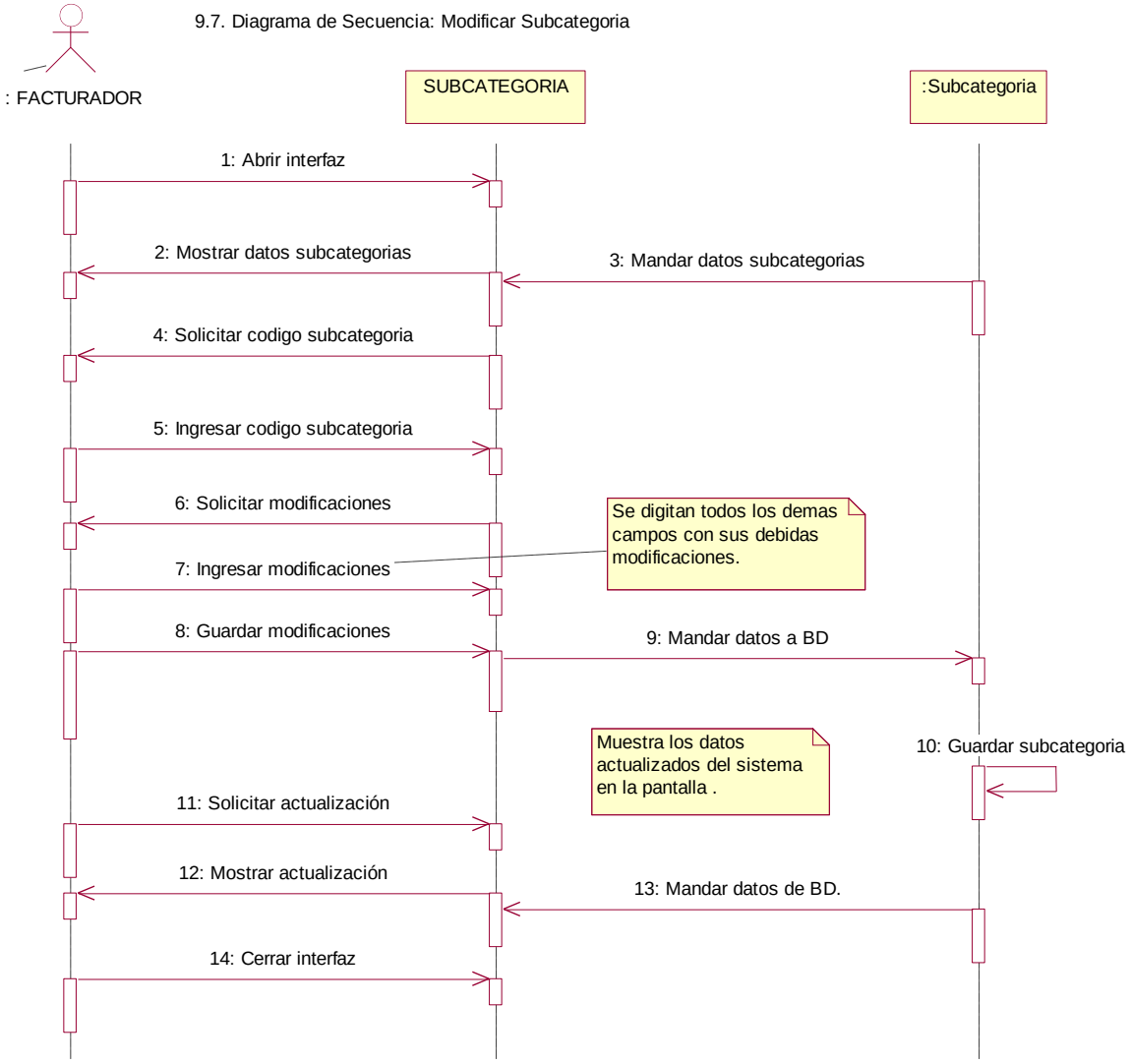



9.6. Diagrama de Colaboración: Modificar Categoria

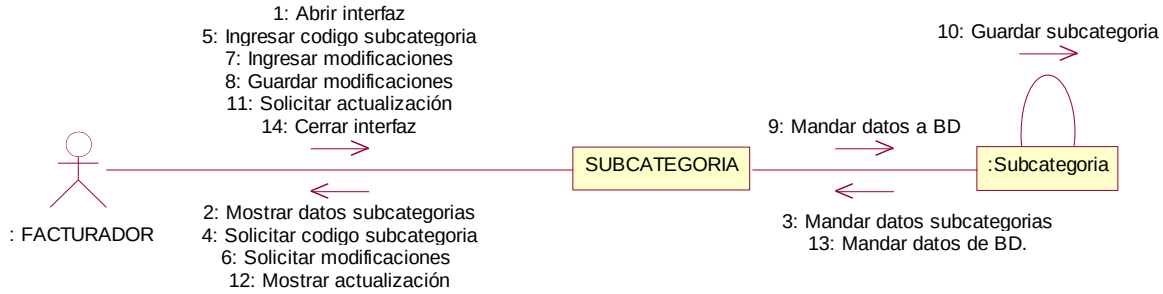


## Escenario 7: Modificar Subcategoría

9.7. Diagrama de Secuencia: Modificar Subcategoria

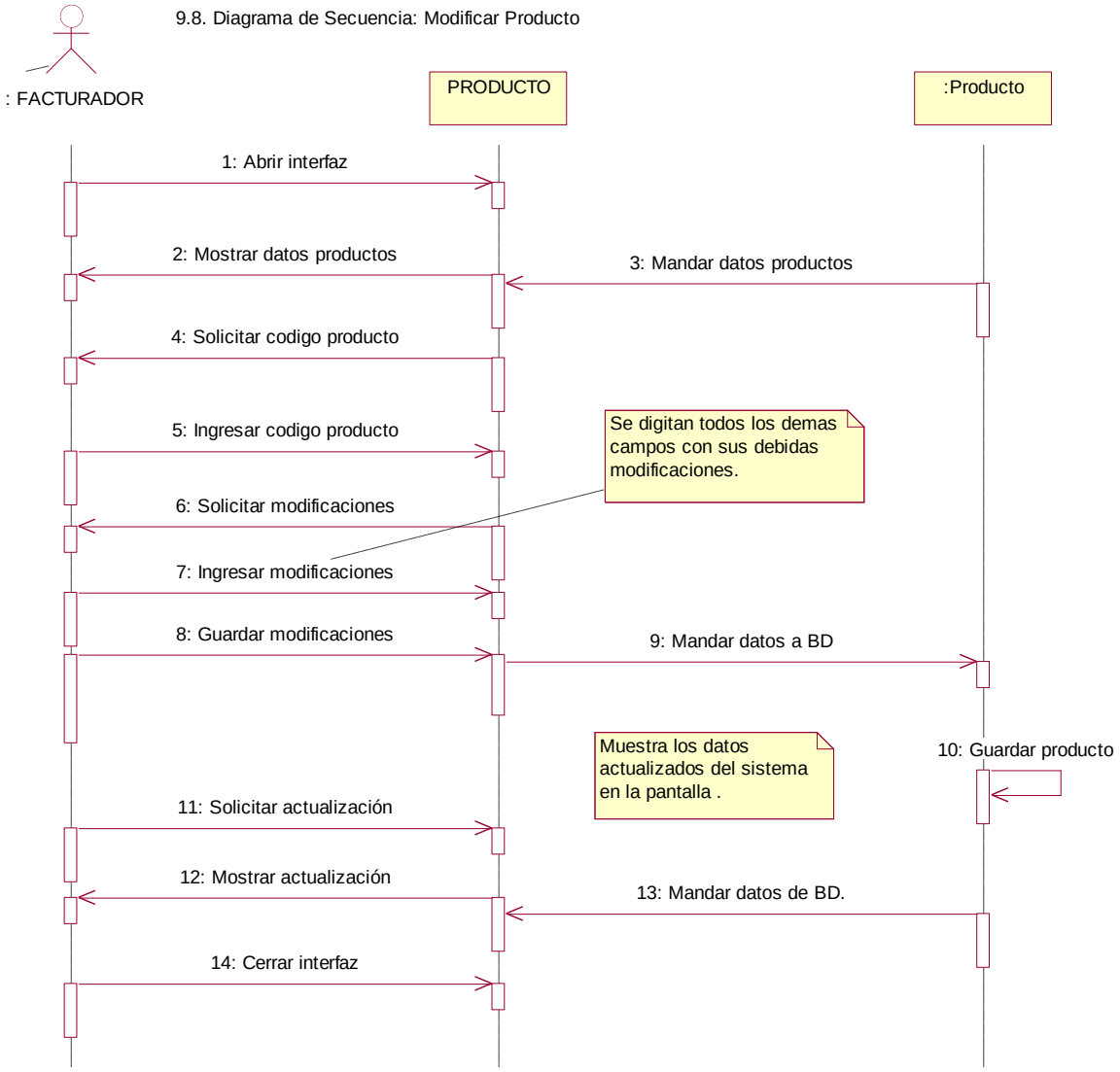



9.7. Diagrama de Colaboración: Modificar Subcategoria

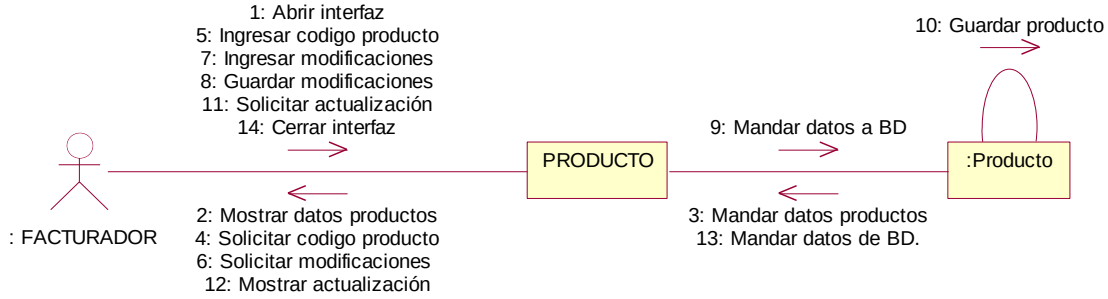


## Escenario 8: Modificar Producto

9.8. Diagrama de Secuencia: Modificar Producto

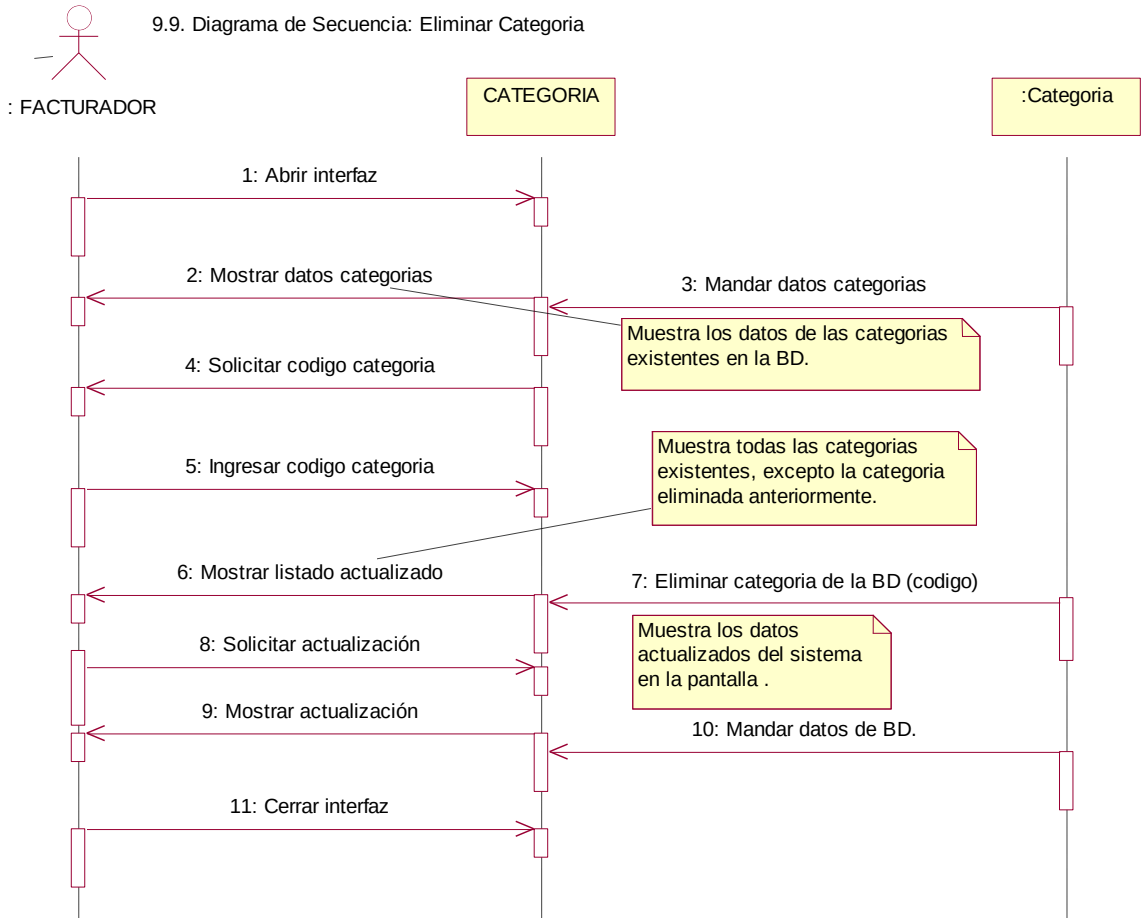



9.8. Diagrama de Colaboración: Modificar Producto

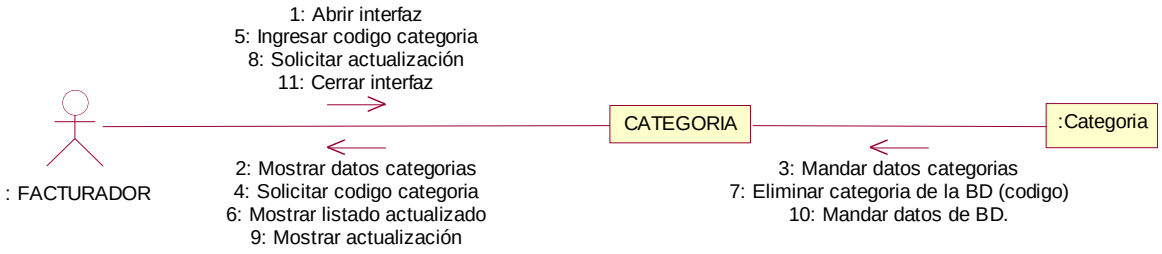


## Escenario 9: Eliminar Categoría

9.9. Diagrama de Secuencia: Eliminar Categoria

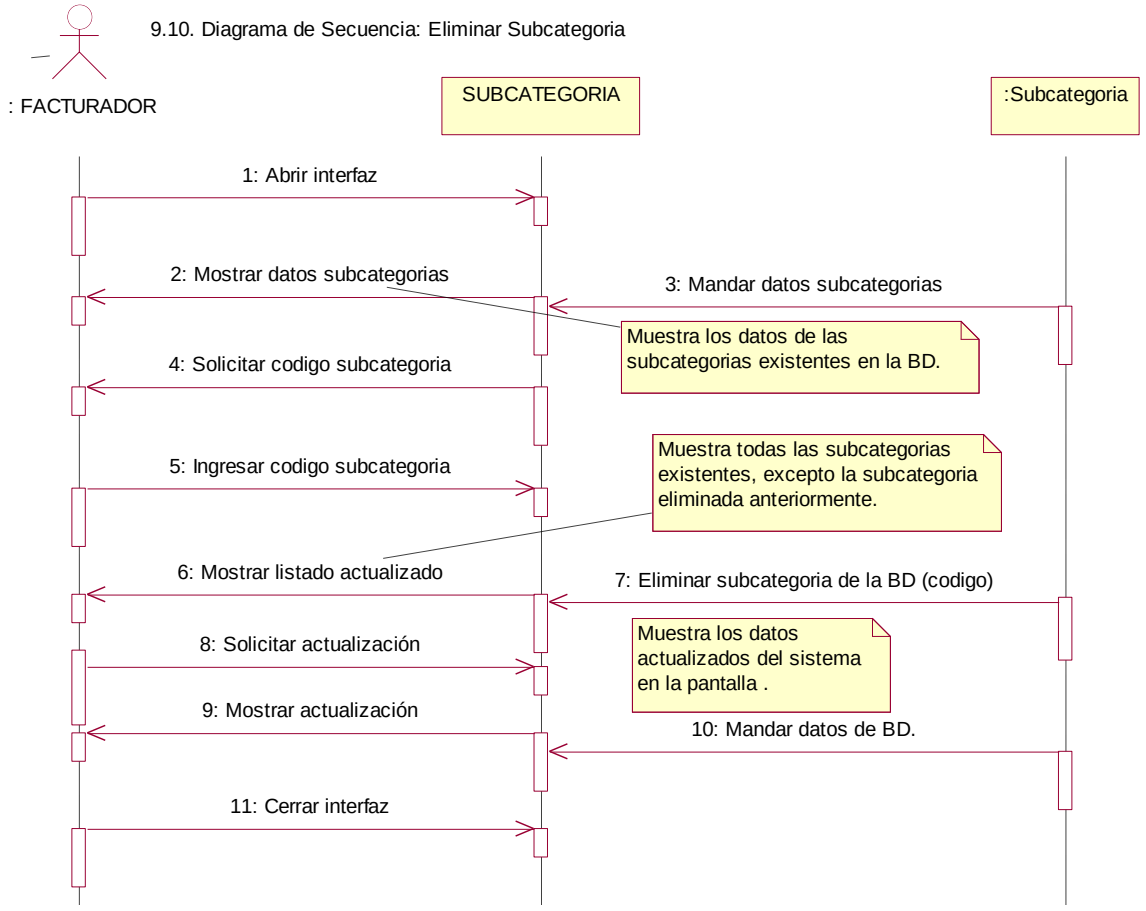



9.9. Diagrama de Colaboración: Eliminar Categoria

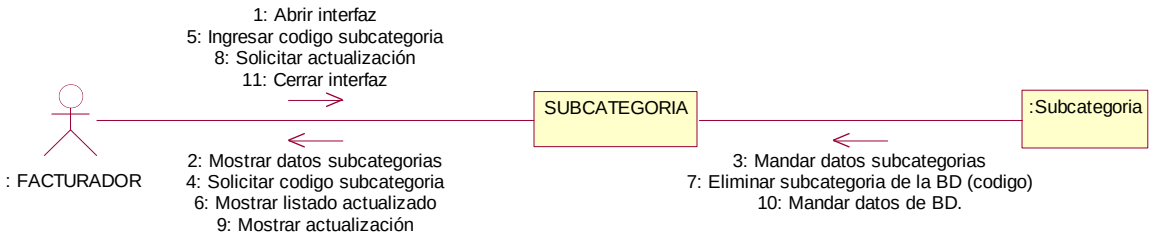


## Escenario 10: Eliminar Subcategoría

9.10. Diagrama de Secuencia: Eliminar Subcategoria

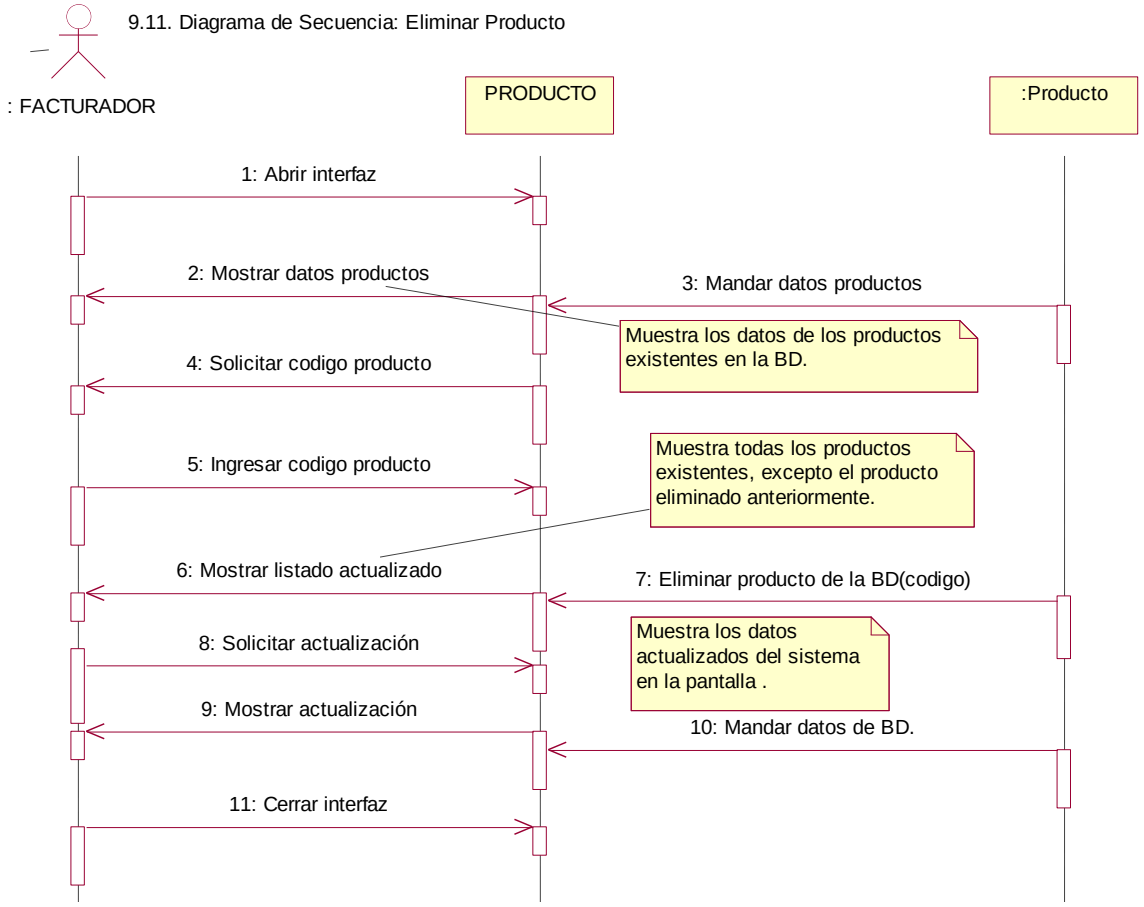



9.10. Diagrama de Colaboración: Eliminar Subcategoria

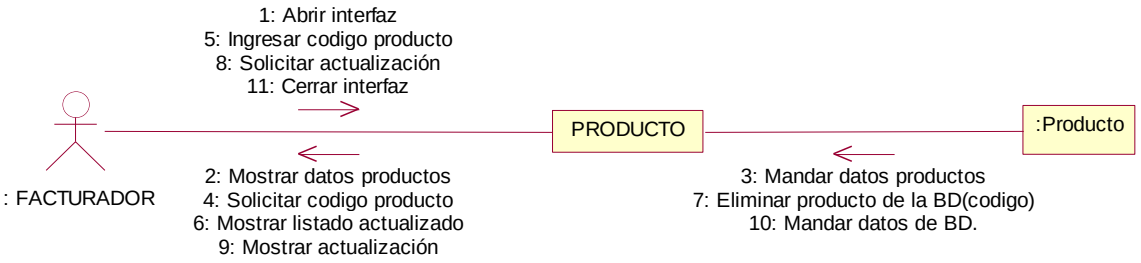


## Escenario 11: Eliminar Producto

9.11. Diagrama de Secuencia: Eliminar Producto



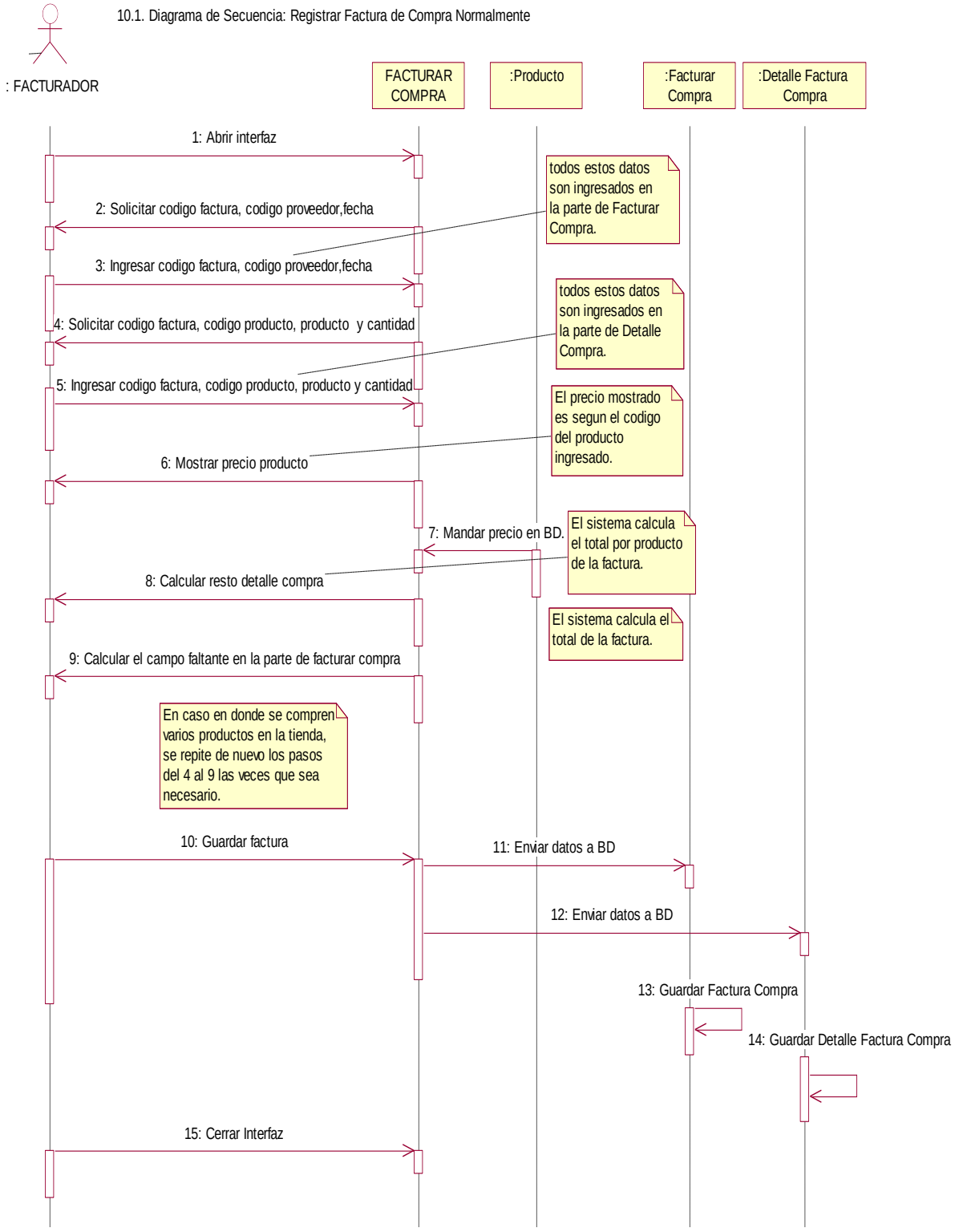

9.11. Diagrama de Colaboración: Eliminar Producto



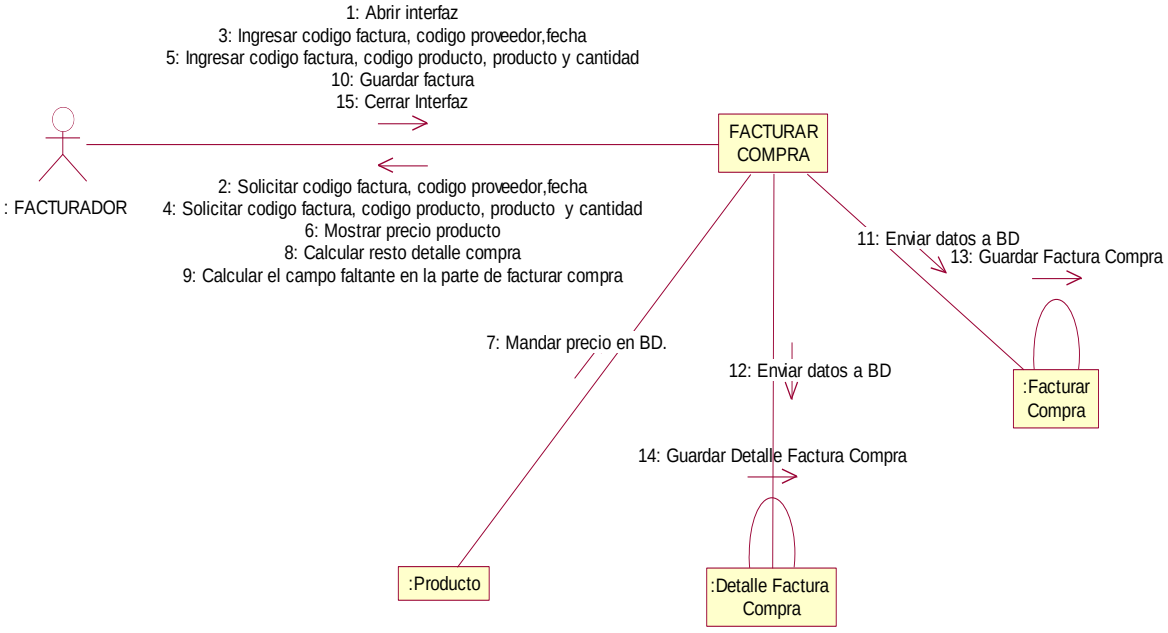
## Caso de Uso 10: Gestionar Compra Proveedor

## Escenario 1: Registrar Factura de Compra Normalmente

10.1. Diagrama de Secuencia: Registrar Factura de Compra Normalmente

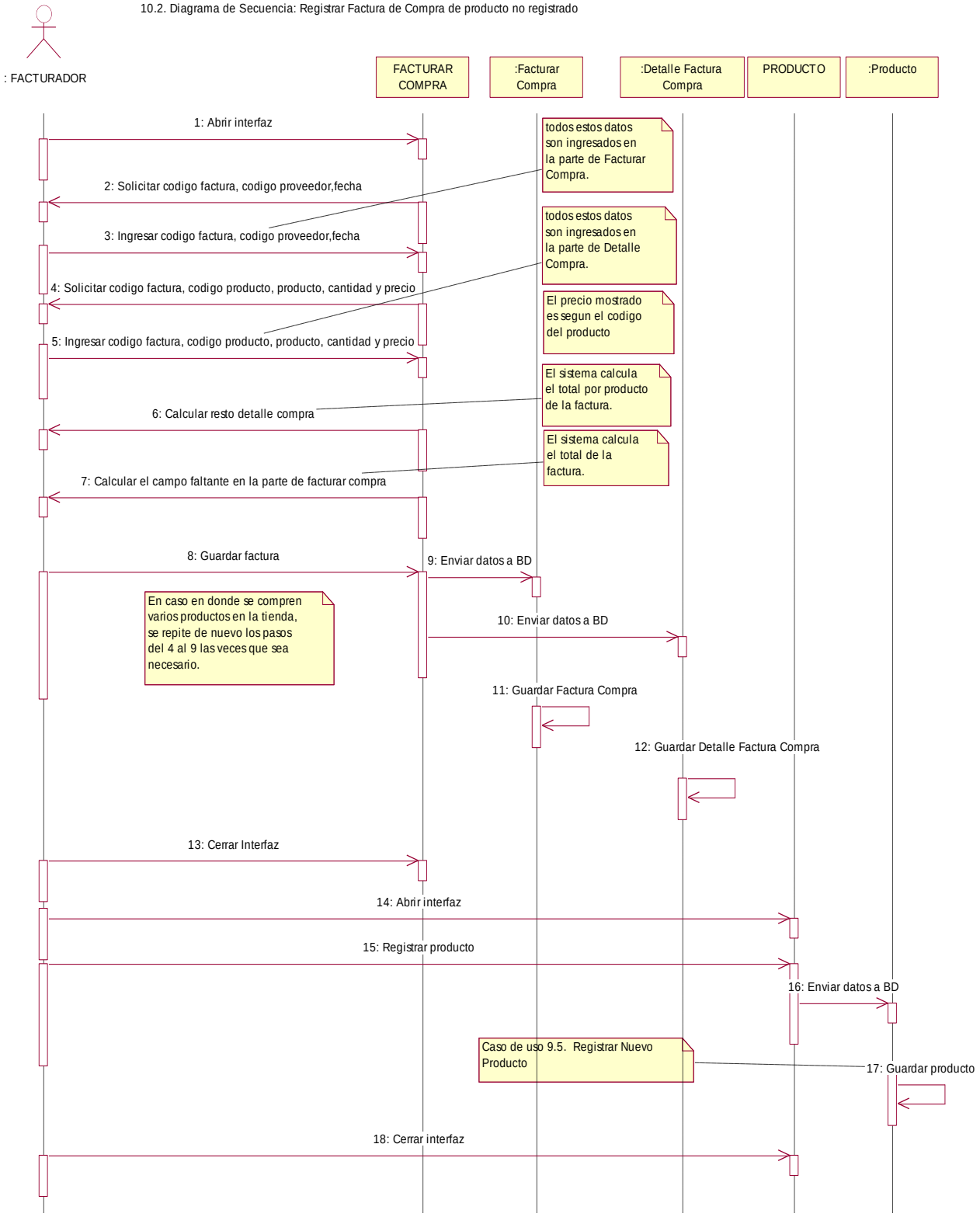


10.1. Diagrama de Colaboración: Registrar Factura de Compra Normalmente

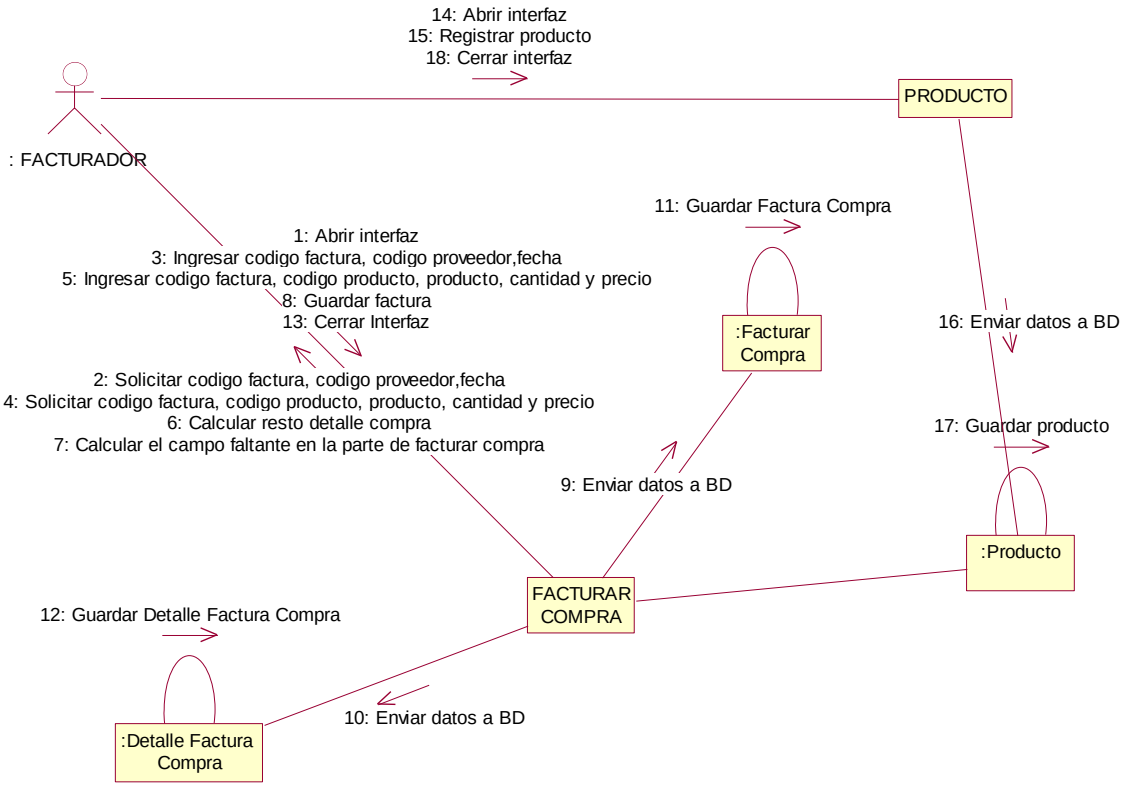


## Escenario 2: Registrar Factura de Compra de producto no registrado

10.2. Diagrama de Secuencia: Registrar Factura de Compra de producto no registrado

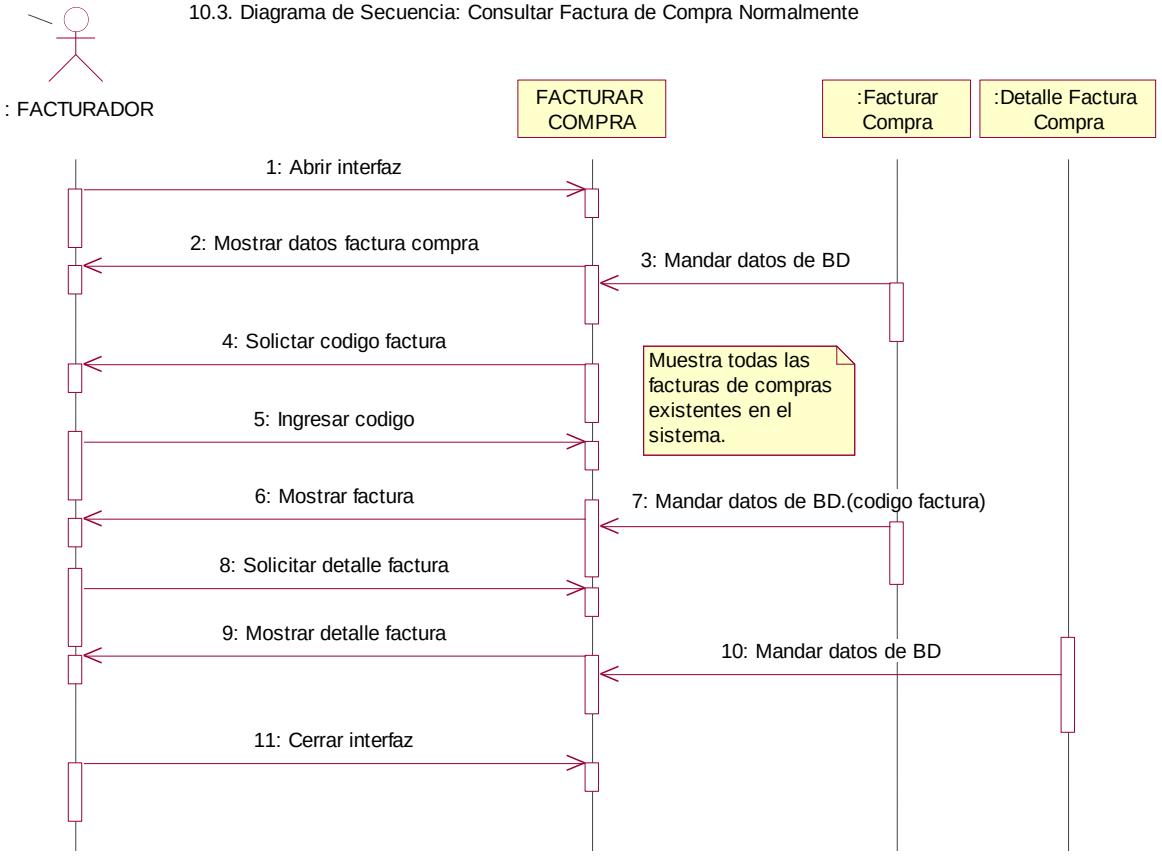


10.2. Diagrama de Colaboración: Registrar Factura de Compra de producto no registrado

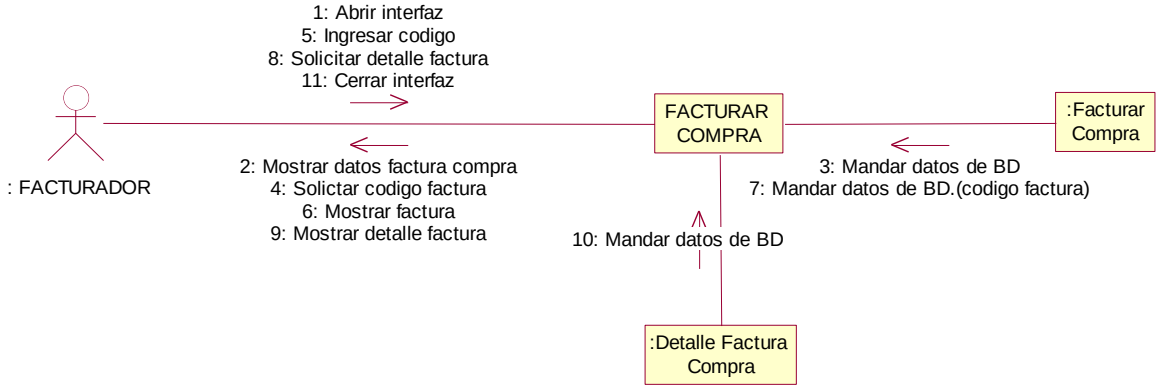


## Escenario 3: Consultar Factura de Compra Normalmente

10.3. Diagrama de Secuencia: Consultar Factura de Compra Normalmente

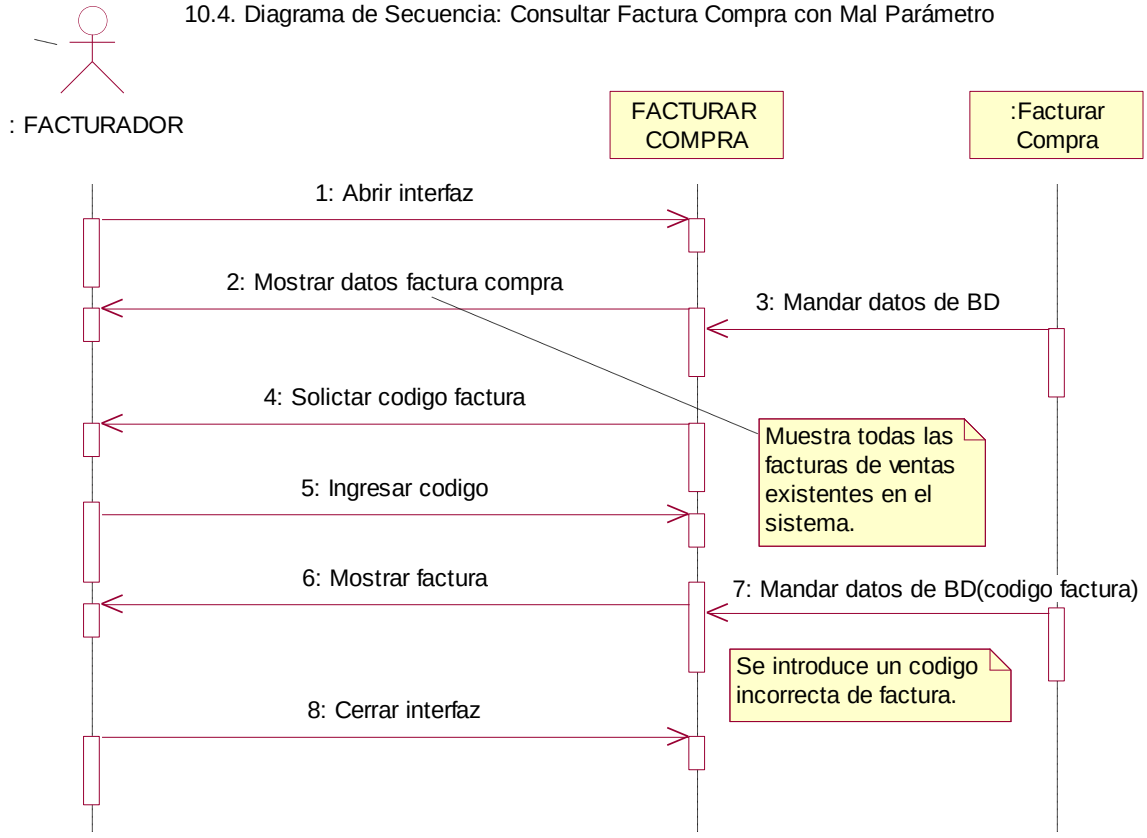



10.3. Diagrama de Colaboración: Consultar Factura de Compra Normalmente

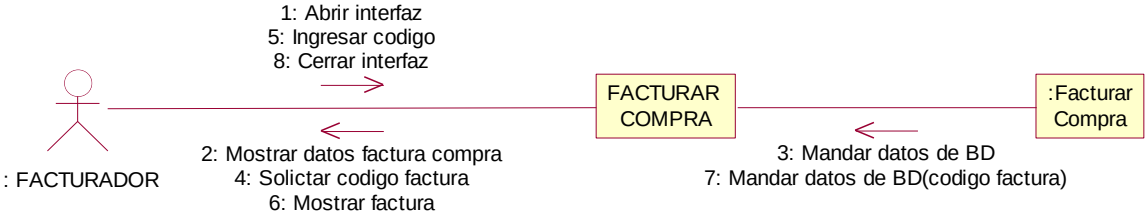


## Escenario 4: Consultar Factura Compra con Mal Parámetro

10.4. Diagrama de Secuencia: Consultar Factura Compra con Mal Parámetro

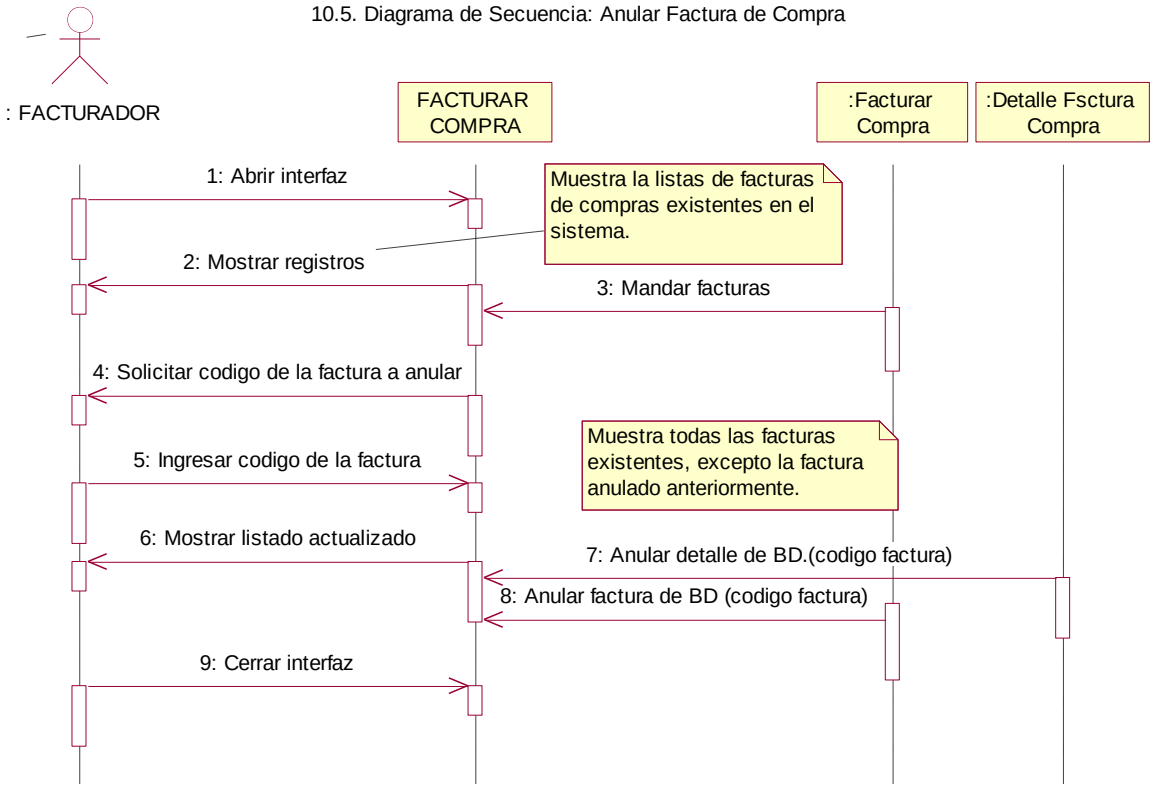



10.4. Diagrama de Colaboración: Consultar Factura Compra con Mal Parámetro

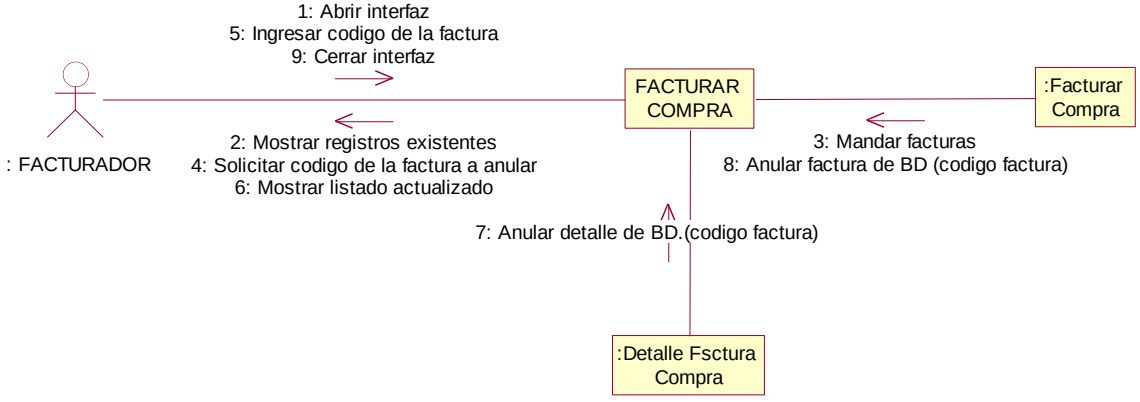


## Escenario 5: Anular Factura de Compra

10.5. Diagrama de Secuencia: Anular Factura de Compra




10.5. Diagrama de Colaboración: Anular Factura de Compra



# III DISEÑO

# 1. DIAGRAMA DE CLASES

Se desea desarrollar un sistema que facture y controle el inventario. El sistema deberá cumplir con las siguientes funciones: controlar las ventas a los clientes, controlar las compras a proveedores, registrar estado de existencia de los productos en la tienda (control de inventario), realizar arqueos diarios en la tienda. El sistema funcionará de la siguiente forma:

Una persona llega a la tienda y comienza a observar todos los productos de la misma, si un producto es de su agrado el se acerca donde el facturar para así poderle hacerle una factura de venta. La factura de venta deberá llevar los siguientes datos: descripción del producto (categoría, subcategoría, tamaño y marca), fecha, nombre del cliente, cantidad, precio unitario, subtotal, descuento, IVA, total, pago y cambio.

El Facturador deberá ser capaz de manipular las facturas de ventas, es decir, hacer los movimientos principales de todo sistema de facturación y control de inventario, los cuales son: Introducir, Modificar, Consultar, Anular. Este mismo podrá emitir una serie de informes del estado de los productos; los cuales son: productos más vendidos, productos menos vendidos, productos agotados y existencia total de productos.

A la hora de la compra a proveedores el propietario deberá recibir un listado de los productos agotados por parte del Faturador para así poder solicitar el pedido al proveedor, el cual estará registrado en el sistema. Los datos relevantes del proveedor serán: nombre proveedor, dirección, teléfono.

El sistema llevara un registro de todas las compras a proveedores y podrá emitir un recibo de compra al mismo (Proveedor). Los datos relevantes en este registro serán: nombre del proveedor, fecha compra, precio unitario, subtotal y total factura. El facturar podrá realizar los movimientos principales mencionado en la venta a los clientes.

El Facturador podrá manipular el catalogo de los productos y el registro de proveedores, es decir, podrá realizar los movimientos principales antes mencionados. A la hora de almacenar el catalogo los productos estarán detallados de siguiente manera: existencia, precio compra, precio venta y a su vez categorizado de dos formas:

- ✓ Por detalle categoría (camisa dama, camisa caballero, pantalón dama, pantalón caballero, calzado dama, calzado caballero).

- ✓ Por detalle producto: Donde estará una subcategoría (adulto, niño), tamaño y marca.

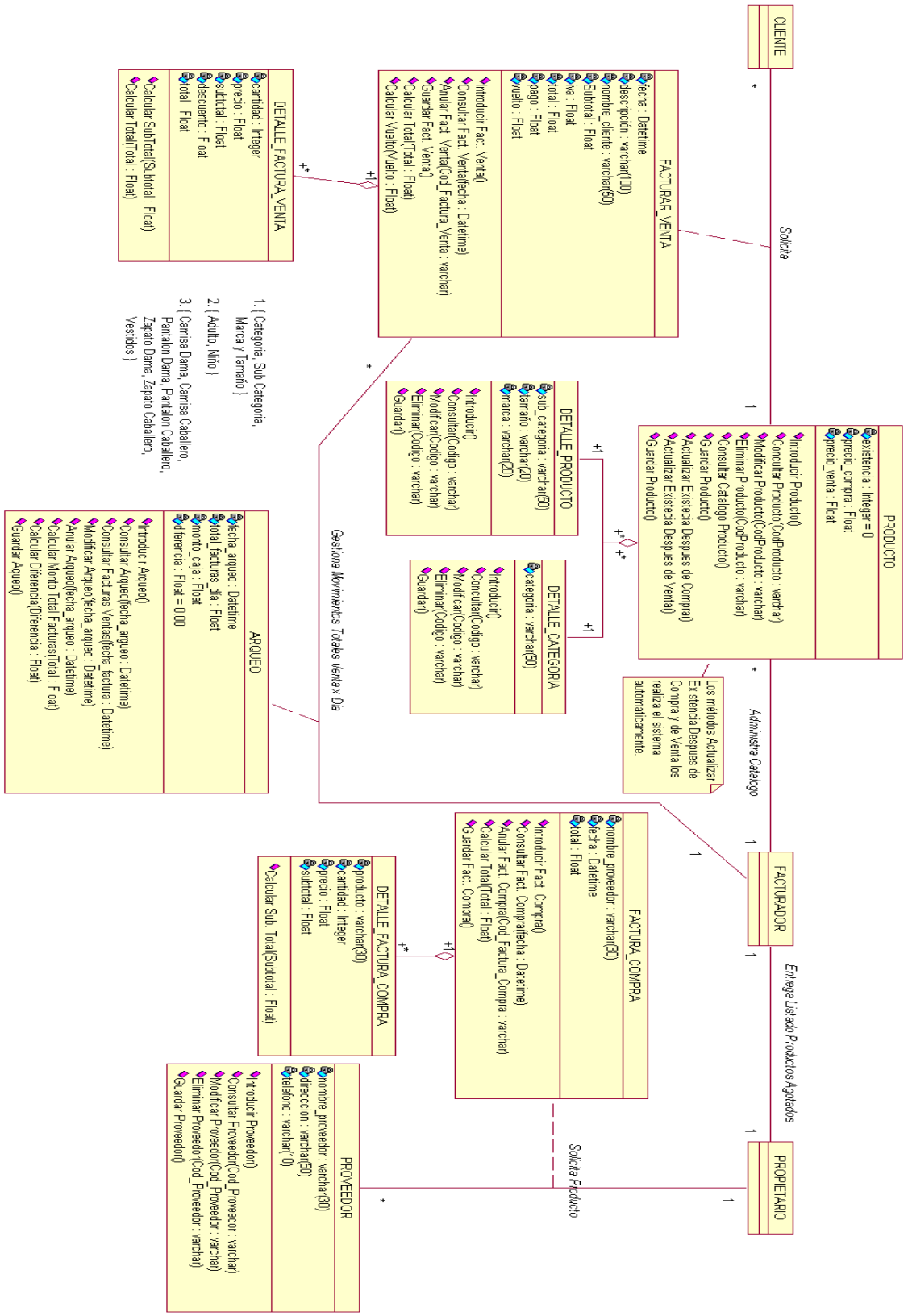
Se podrá introducir, eliminar, modificar y consultar cualquier categoría, marca o tamaño en el catalogo. Todo esto lo va a realizar el Facturador.

A la hora de realizar el arqueo el sistema podrá mandar llamar el monto total de las facturas de venta del día para así compararlo con caja chica y así ver si no hay diferencia entre ambos. El sistema llevara un control de arquero diarios el cual llevara los siguientes datos: fecha arqueo, total facturas por día, monto caja chica, diferencia. También se podrá introducir, consultar, anular y modificar cualquier arqueo registrado del sistema.

Por ultimo el sistema será capaz emitir un listado de los proveedores, informe de arqueo por día el cual se le entregara al propietario, deberá imprimir todos los productos con sus precios de ventas (etiquetas) y el estado del producto para su debida marcación (Descuento, promoción, precio normal).

Todo lo antes mencionados son las funciones mas relevantes que hará el Sistema de Facturación y Control de Inventario en la tienda DENMAR.

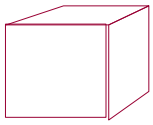
## DIAGRAMA DE CLASES



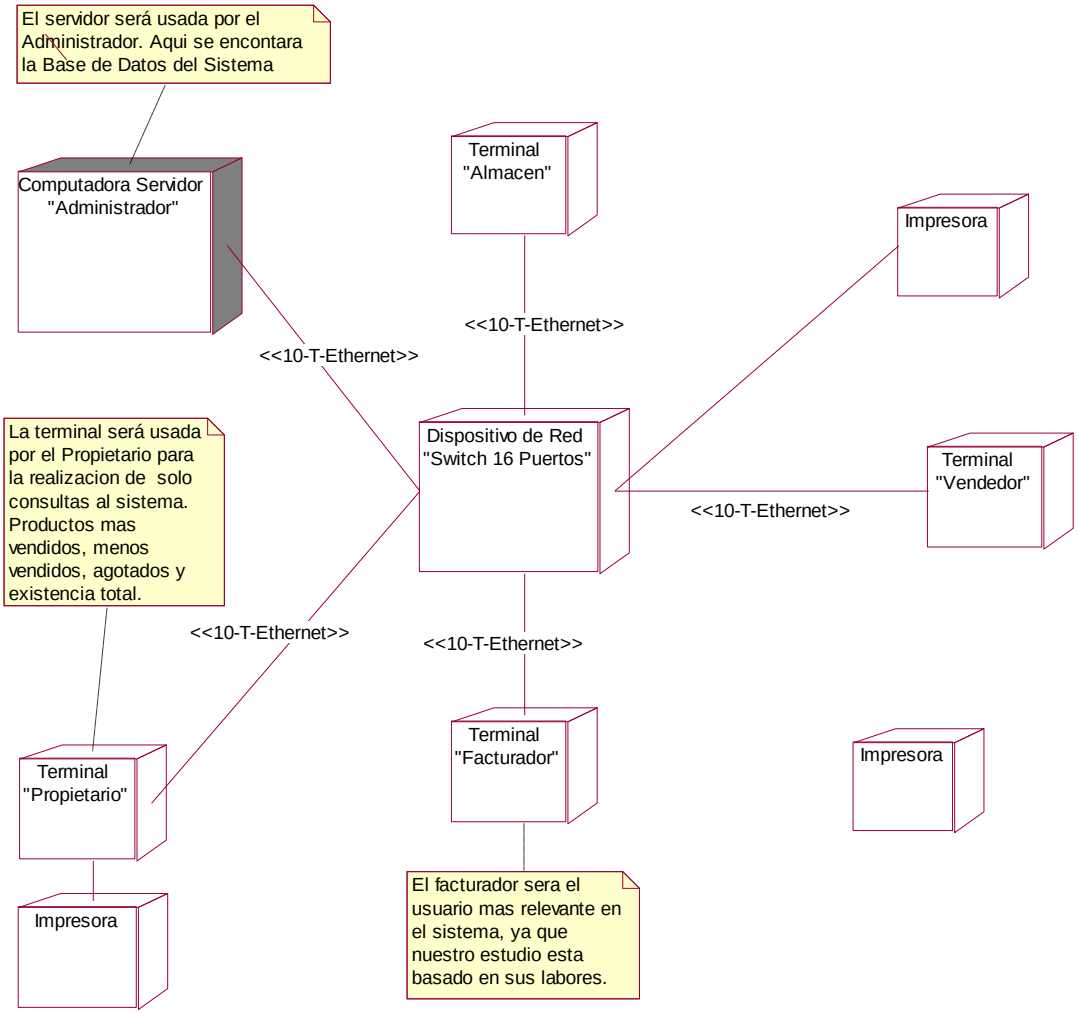
## 2. DIAGRAMA DE DESPLIEGUE

Debido que en la actualidad en la tienda todo se hace manualmente, no poseen una arquitectura actual de componentes, por lo cual se representa por un nodo en blanco.

Distribución Actual



Distribución Propuesta



# 3. MODELO DE DATOS

**Proceso de Normalización**

El interés de esta empresa es manejar de forma detallada la compra y venta de sus productos, esto se refiere a la facturación y control de inventario, lo que implica manejar de forma detallada los datos de las compras, de las ventas y de los productos.

La forma de trabaja de la tienda DENMAR es la siguiente:

- ✓ La compra de los repuestos al proveedor, en lo cual le interesa tener todos los datos generales de los proveedores y facturar esa comprar de productos para llevar un control de las mismas.
- ✓ La venta de los productos al cliente, en lo cual le interesa facturar la venta de los productos para llevar un control de las mismas.
- ✓ Llevar un control de inventario en la tienda.
- ✓ Realizar los arqueos diarios en la tienda.

Los datos que se necesitan conocer para llevar una adecuada facturación y control de inventario es el siguiente:

**PROVEEDOR**

Nombre_Proveedor

Dirección

Teléfono

**COMPRA PRODUCTOS**

Nombre_Proveedor

Fecha

Productos

Cantidad

Precio

Subtotal

Total

**VENTA PRODUCTOS**

Fecha

Descripción

Nombre_Cliente

Cantidad

Precio_Unitario

Subtotal

Descuento

Total

IVA

Total

Pago

Vuelto

**PRODUCTOS**

Existencia

Precio_Compra

Precio_Venta

Categoría

Subcategoría

Tamaño

Marca

**ARQUEO**

Fecha_Arqueo

Total_Facturas_Día

Monto_Caja

Diferencia

**Primera Forma Normal**

El primer paso es eliminar todas las columnas o campos repetidos. De acuerdo con la regla a las columnas repetidas se le debe crear su propia tabla.

De lo anterior resulta lo siguiente:

| **PRODUCTO**<br>**Nombre del Campo** | **FACTURAR_COMPRA**<br>**Nombre del Campo** | **FACTURAR_VENTA**<br>**Nombre del Campo** |
|---|---|---|
| Existencia | Nombre_Proveedor | Fecha |
| Precio_Compra | Fecha | Descripción |
| Precio_Venta | Producto | Nombre_Cliente |
| Categoría | Cantidad | Cantidad |
| Subcategoría | Precio | Precio_Unitario |
| Tamaño | Subtotal | Subtotal |
| Marca | Total | Descuento |
| | | IVA |
| | | Total |
| | | Pago |
| | | Vuelto |

| **ARQUEO**<br>**Nombre del Campo** | **PROVEEDOR**<br>**Nombre del Campo** |
|---|---|
| Fecha_Arqueo | Nombre_Proveedor |
| Total_Facturas_Día | Dirección |
| Monto_Caja | Teléfono |
| Diferencia | |

**Segunda Forma Normal**

Después de aplicar la primera forma normal resultaron 5 tablas. Pero hay un problema, no hay como relacionar las tablas, entonces el siguiente paso en añadir un campo calve a cada una de las tablas para que se establezca la relación. La solución a esto es la asignación de ID a las tablas estableciendo las relaciones necesarias.

Las tablas de la base de datos quedan de la siguiente manera:

| **PRODUCTO** | **FACTURAR_COMPRA** | **FACTURAR_VENTA** |
|---|---|---|
| **Nombre del Campo** | **Nombre del Campo** | **Nombre del Campo** |
| Cod_Producto | Cod_Factura_Compra | Cod_Factura_Venta |
| Cod_Categoria | Cod_Proveedor | Cod_Producto |
| Cod_Subcategoria | Cod_Producto | Fecha |
| Existencia | Nombre_Proveedor | Descripción |
| Precio_Compra | Fecha | Nombre_Cliente |
| Precio_Venta | Producto | Cantidad |
| Categoría | Cantidad | Precio_Unitario |
| Subcategoría | Precio | Subtotal |
| Tamaño | Subtotal | Descuento |
| Marca | Total | IVA |
| | | Total |
| | | Pago |
| | | Vuelto |

| **ARQUEO** | **PROVEEDOR** |
|---|---|
| **Nombre del Campo** | **Nombre del Campo** |
| Cod_Arqueo | Cod_Proveedor |
| Fecha_Arqueo | Nombre_Proveedor |
| Total_Facturas_Día | Dirección |
| Monto_Caja | Teléfono |
| Diferencia | |

**Tercera Forma Normal**

La regla de la Tercera Forma Normal señala que hay que eliminar y separar cualquier dato que no sea clave. El valor de esta columna debe depender de la clave. Todos los valores deben identificarse únicamente por la clave. Todos los campos se necesitan en todas las tablas, esto debido a todas las especificaciones que quiere el dueño del negocio. Se separan alguna tablas por quedaran muy sobrecargadas de información. Las tablas de la base de datos quedan de la siguiente manera:

| FACTURAR_COMPRA | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Cod_Factura_Compra | Varchar(15) |
| Cod_Proveedor | Varchar(15) |
| Fecha_Compra | Datetime |
| Total | Float |

| DETALLE_FACTURA_COMPRA | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Id_Detalle_Compra | Varchar(15) |
| Cod_Factura_Compra | Varchar(15) |
| Cod_Producto | Varchar(15) |
| Producto | Varchar(100) |
| Cantidad | Integer |
| Precio | Float |
| Subtotal | Float |

| FACTURAR_VENTA | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Cod_Factura_Venta | Varchar(15) |
| Fecha_Venta | Datetime |
| Descripción | Varchar(100) |
| Nombre_Cliente | Varchar(50) |
| Subtotal | Float |
| IVA | Float |
| Total | Float |
| Pago | Float |
| Vuelto | Float |

| DETALLE_FACTURA_VENTA | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Id_Detalle_Venta | Varchar(15) |
| Cod_Factura_Venta | Varchar(15) |
| Cod_Producto | Varchar(15) |
| Cantidad | Integer |
| Precio | Float |
| Subtotal | Float |
| Descuento | Float |
| Total | Float |

| PRODUCTO | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Cod_Producto | Varchar(15) |
| Cod_Categoria | Varchar(15) |
| Cod_Subcategoria | Varchar(15) |
| Precio_Compra | Float |
| Precio_Venta | Float |
| Existencia | Integer |

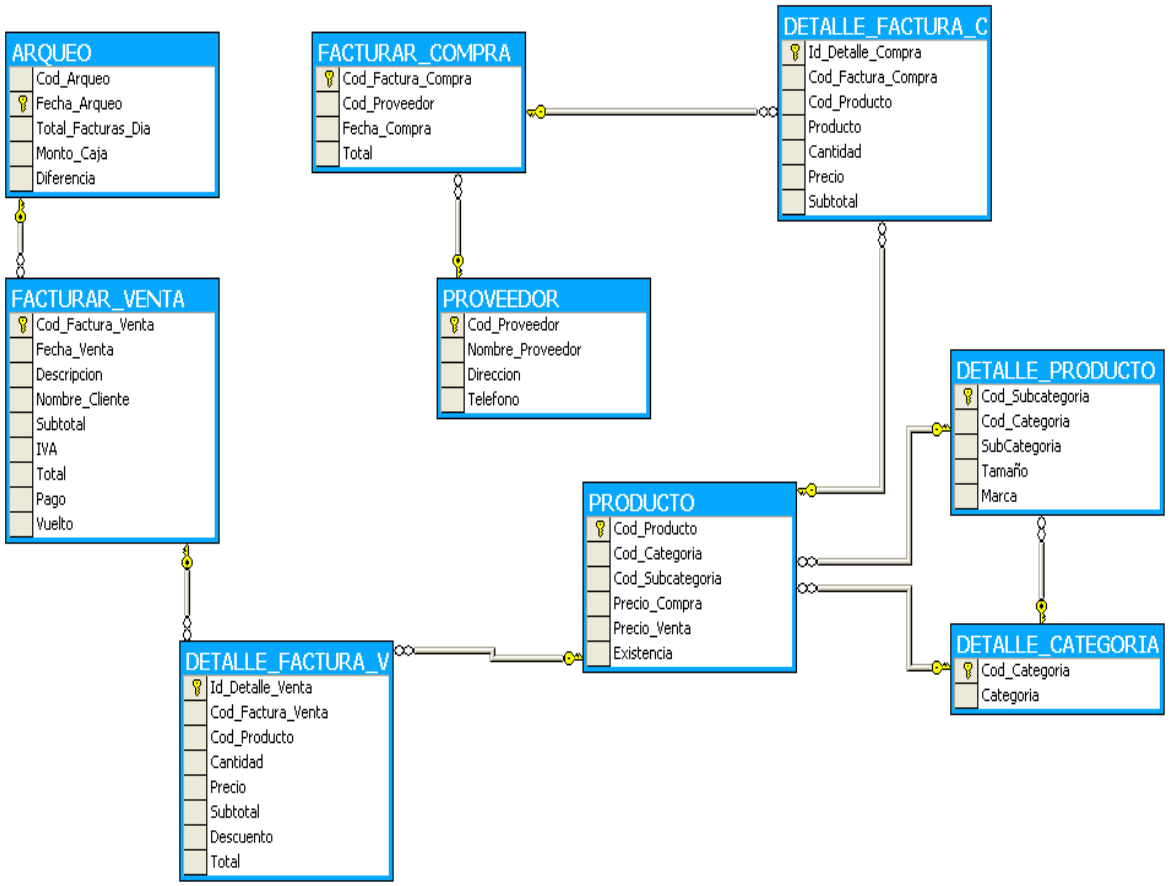
| DETALLE_CATEGORIA | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Cod_Categoria | Varchar(15) |
| Categoría | Varchar(50) |

| DETALLE_PRODUCTO | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Cod_Subcategoria | Varchar(15) |
| Cod_Categoria | Varchar(15) |
| Subcategoría | Varchar(50) |
| Tamaño | Varchar(20) |
| Marca | Varchar(20) |

| ARQUEO | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Cod_Arqueo | Varchar(15) |
| Fecha_Arqueo | Datetime |
| Total_Facturas_Día | Float |
| Monto_Caja | Float |
| Diferencia | Float |

| PROVEEDOR | |
|---|---|
| **Nombre del Campo** | **Tipo Campo** |
| Cod_Proveedor | Varchar(15) |
| Nombre_Proveedor | Varchar(30) |
| Dirección | Varchar(50) |
| Teléfono | Varchar(10) |

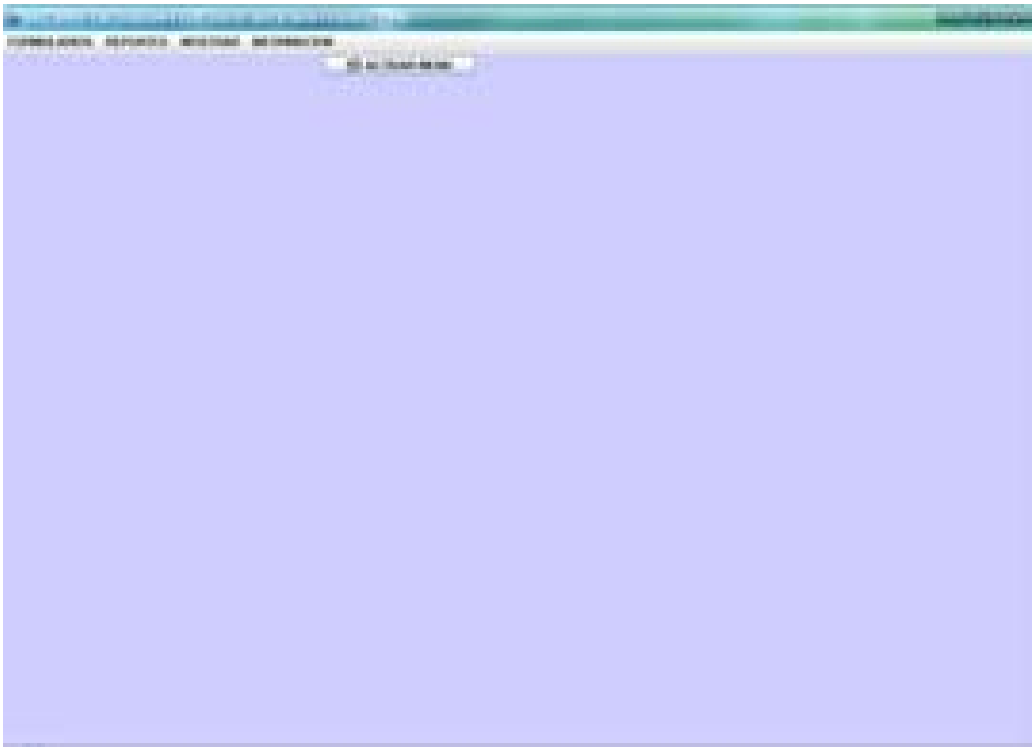
## TABLAS RELACIONADAS



# 4. PANTALLAS DEL SISTEMA

## USUARIO


## PRINCIPAL



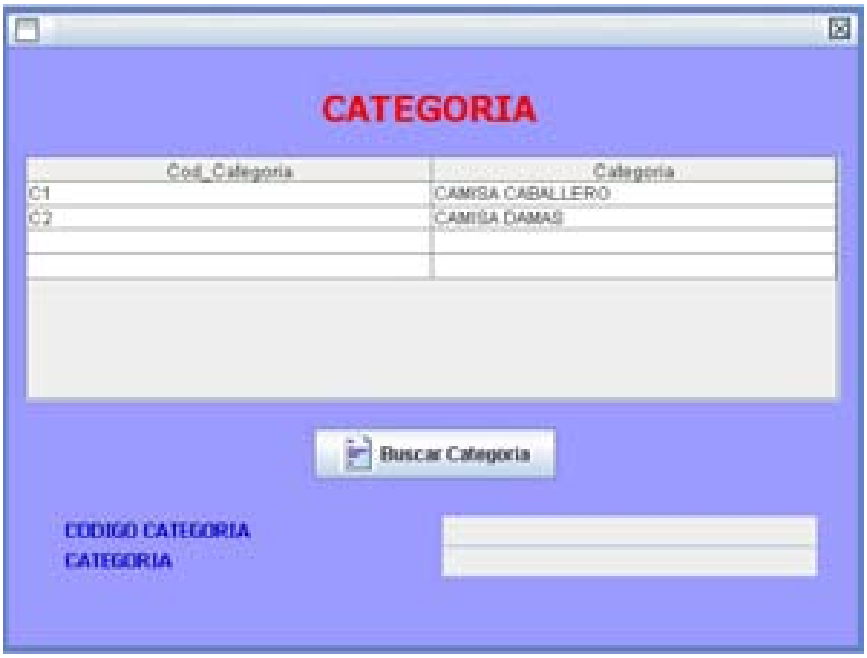
## ARQUEOS


ARQUEO
Cod_Arqueo
Fecha_Arqueo
Total_Facturas
Monto_Caja
Diferencia
A1
621
621
0
Mostrar Facturas
Cod_Facturas
Fecha_Venta
Descripcion
Nombre_Cte
Subtotal
IVA
Total
Pago
Vuelto


ARQUEO
Cod_Arqueo
Fecha_Arqueo
Total_Facturas
Monto_Caja
Diferencia
A1
2008-02-06 00:0...
621
621
0
Buscar Arqueo
Salir
CODIGO ARQUEO
FECHA ARQUEO
TOTAL FACTURAS
MONTO CAJA
DIFERENCIA

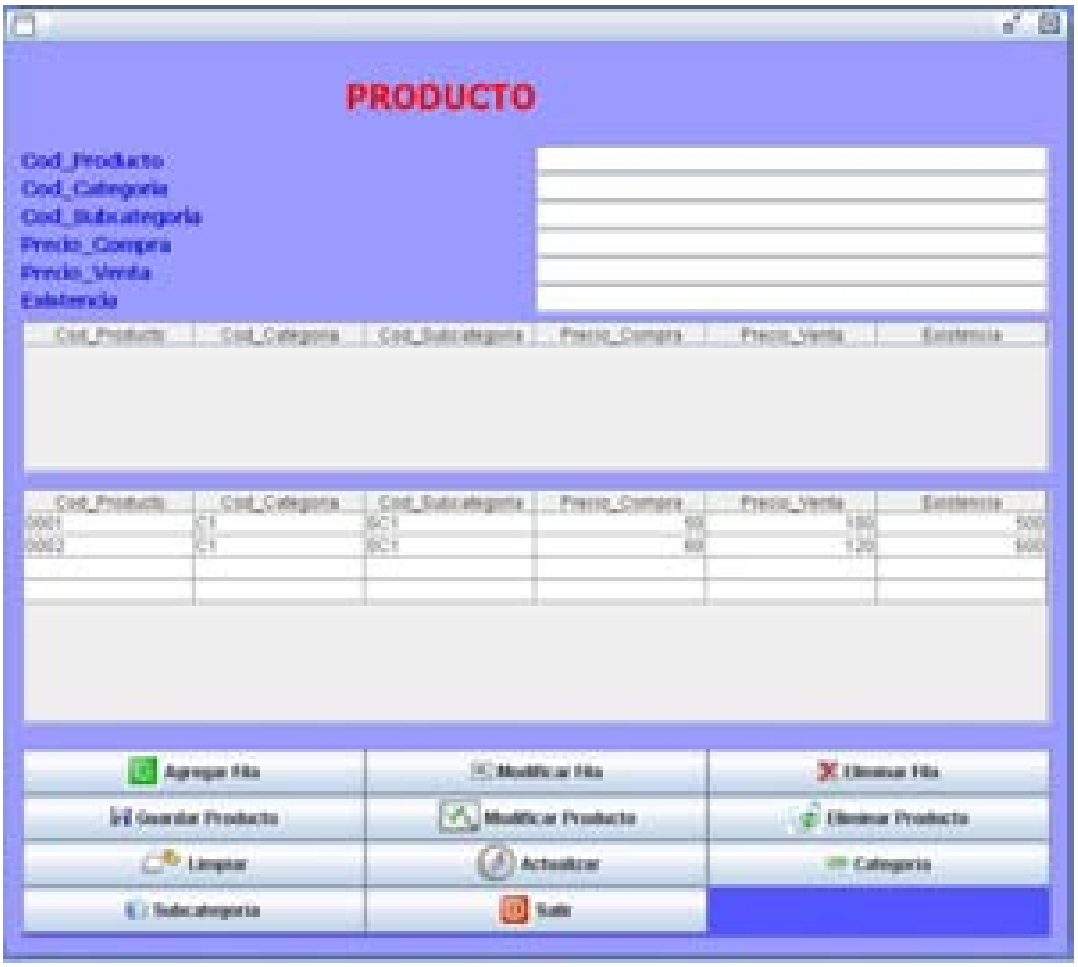
## CATEGORIA

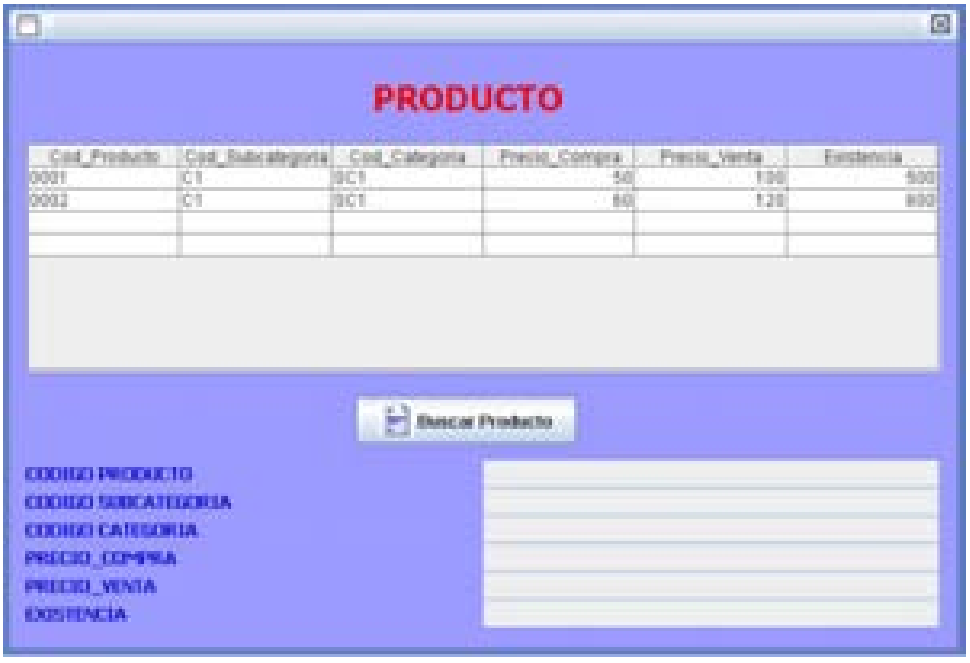

CATEGORIA
Cod_Categoria
Categoria
Cod_Categoria
Categoria
Cod_Categoria
Categoria
C1
CAMISA CABALLERO
C2
CAMISA DAMAS
Agregar Fila
Modificar Fila
Eliminar Fila
Guardar Categoria
Modificar Categoria
Eliminar Categoria
Limpiar
Actualizar
Salir


CATEGORIA
Cod_Categoria
Categoria
C1
CAMISA CABALLERO
C2
CAMISA DAMAS
Buscar Categoria
CODIGO CATEGORIA
CATEGORIA

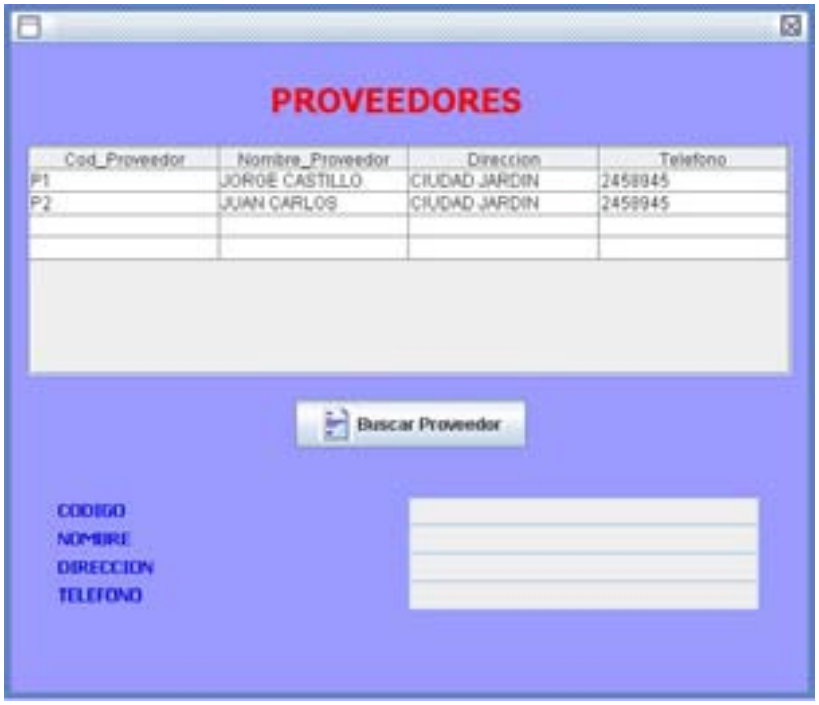
## SUBCATEGORIA


**PRODUCTO**


PRODUCTO
Cod_Producto
Cod_Categoria
Cod_Subcategoria
Precio_Compra
Precio_Venta
Existencia
Cod_Producto
Cod_Categoria
Cod_Subcategoria
Precio_Compra
Precio_Venta
Existencia
0001
C1
SC1
50
100
500
0002
C1
SC1
60
120
600
Agregar Fila
Modificar Fila
Eliminar Fila
Guardar Producto
Modificar Producto
Eliminar Producto
Limpiar
Actualizar
Categoria
Subcategoria
Salir


PRODUCTO
Cod_Producto
Cod_Subcategoria
Cod_Categoria
Precio_Compra
Precio_Venta
Existencia
0001
C1
SC1
50
100
500
0002
C1
SC1
60
120
600
Buscar Producto
CODIGO PRODUCTO
CODIGO SUBCATEGORIA
CODIGO CATEGORIA
PRECIO_COMPRA
PRECIO_VENTA
EXISTENCIA

## PROVEEDORES

## VENTAS


FACTURAR VENTA
Cod_Factura
Descripcion
Subtotal
Total
Vuelto
Fecha
Nombre_Cliente
IVA
Pago
Calcular Total Factura
Calcular Vuelto
AGREGAR FILA
Salir
DETALLE VENTA
Cod_Factura_Venta
Cantidad
Subtotal
Total
Cod_Producto
Precio
Descuento
Calcular
AGREGAR FILA
ELIMINAR FILA
Guardar Factura
Anular Factura


FACTURAS DE VENTA

## COMPRAS


FACTURAR COMPRA
Cod_Factura
Cod_Proveedor
Fecha
Total
Calcular Total Factura
Cod_Factura
Cod_Producto
Fecha
Total
Salir
DETALLE COMPRA
Cod_Factura_Venta
Producto
Precio
Cod_Producto
Cantidad
Subtotal
Calcular
Cod_Factura_Venta
Cod_Producto
Producto
Cantidad
Precio
Total
Guardar Factura


FACTURAS DE COMPRA
Cod_Factura
Cod_Proveedor
Fecha_Compra
Total
Buscar Factura
Mostrar Detalle
Salir
TOTAL
Cod_Factura_Compra
Cod_Producto
Producto
Cantidad
Precio
Subtotal

# IV DISEÑO DE RED

# 1. Análisis de Requerimientos

## 1.1. Capturar y Listar Requerimientos

### 1.1.1. Condiciones Iníciales

#### 1.1.1.1. Tipo del Proyecto

Actualmente la Tienda DENMAR carece de una estructura de red obviamente debido a la falta de computadoras que puedan ser utilizadas para las labores en la misma. Por tanto nuestro tipo de proyecto según las condiciones iníciales será la propuesta de una nueva red.

#### 1.1.1.2. Ámbito del Diseño

La tienda DENMAR abarca un área de 30 $mts^2$. El número de sitios que conformara la red serán 5 y las distancias con respecto al área administrativa son las siguientes:

- Propietario: Oficina Contigua.
- Facturación: Diez metros de distancia.
- Ventas: Quince metros de distancia.
- Almacén: Veinte metros de distancia.

#### 1.1.1.3. Objetivos Iníciales

- Permitir que el conjunto de equipos (PC y/o dispositivos) conectados compartan las información (Archivos), Recursos (Impresoras, Scanner) y servicios (Internet) que los usuarios crean más relevantes.
- Simplificar el tiempo de respuesta al cliente.

#### 1.1.1.4. Fuerzas Externas

La empresa desea que se establezcan reglas de seguridad particularmente en el departamento de Administración y de Ingeniería. También se desea establecer políticas de seguridad en los accesos a la red pública (Internet).

**Uso de la Red**

- ✓ Los recursos de la red deben ser usados única y exclusivamente para cumplir con las tareas dentro de la tienda.
- ✓ No sobre utilizar la red, existe una capacidad límite del envío de información, hacer esto puede saturar la red y causar problemas.
- ✓ La configuración de la red es prioridad del administrador, no sustraerla ni modificarla.
- ✓ Toda la información que se encuentre en las maquinas debe ser compartida para los demás usuarios de la red.
- ✓ Solamente la maquina servidor puede tener acceso sin restricción a Internet para evitar posibles contagios de virus.
- ✓ Los permisos de acceso a Internet serán otorgados por el administrador por medio del servidor.

**1.2. Desarrollar Métricas de Servicio.**

| **Métricas de Servicio** | **¿Dónde se medirán?** | **Método de medición** |
|---|---|---|
| Medir rendimiento | En la estación que servirá como servidor. (Administrador). | Utilizando el monitoreo de retardo en la red. Haciendo Ping las estaciones clientes. |
| Medir trafico en la red | En el servidor | Utilizaremos la herramienta: **Wireshark,** la cual sirve para monitorear la transferencia de paquetes entre las estaciones en la red. |

## 1.3. Caracterizar el rendimiento

### 1.3.1. Patrones de uso

Sistema de Facturación y Control de Inventario: el número de usuario para esta aplicación serian seis. La frecuencia de uso esperada seria seis sesiones por día, con una duración de 8 horas. Nosotros esperamos tener dos sesiones simultaneas: la sesión del facturador y la del vendedor.
Servicio de Internet: Los usuarios de este servicio serian: El propietario, el administrador.

### 1.3.2. Usuarios Relevantes

Estos serian el administrador, el facturador.

### 1.3.3. Comportamiento de la aplicación

El tamaño de los datos va a ser variable dependiendo de la cantidad de información contenida en la factura a la hora de realizar la transacción, la frecuencia y duración de la transferencia dependerá el tipo de gestión que estén realizando las estaciones.

Toda la información va a ser guardada en el servidor en la aplicación: Microsoft SQL 2000, es decir la dirección del flujo va ha ser cliente <--> servidor.

Las aplicaciones mas relevantes que se van ha usar en la red serán el sistema de facturación y control de inventario.

## 2. Análisis de Flujo

| Dirección Fuente | Dirección Destino | Tipo de Información |
|---|---|---|
| Facturación | Servidor | Datos que contendrá el sistema, los cuales son: La factura de cada venta y compra registrada en la tienda, la información más relevante de los productos existentes en la tienda, la lista de los proveedores.<br>Archivos de texto: Documentación de uso exclusivo en la tienda. |
| Vendedor | Servidor | Verificación información contenida en el sistema. |
| Almacén | Servidor | Control de los datos contenidos en el sistema. Solo en la parte de existencia de los productos. |
| Propietario | Servidor | Visualización de todos los archivos contenidos en la red. |

Tipo de flujo que existirá en la red será el siguiente:

Toda la información se hará almacenada en la estación que se comporte como servidor, es decir el flujo de la información será cliente servidor.

# 3. Diseño Lógico

## 3.1. Establecer objetivos de diseño

Los objetivos primordiales que se quieren alcanzar al hacer el diseño lógico de la red:

1.1. Disminuir lo mas posible los costos de desarrollo y de operación de la red que se esta proponiendo.

1.2. Facilitar el uso de todas las aplicaciones que se van a utilizar en la red.

1.3. Lograr el mayor rendimiento de las aplicaciones en la red.

1.4. Facilitar el uso de todas las aplicaciones que se compartirán en la red.

1.5. Proponer un diseño de red no muy complejo para así facilitar su administración.

1.6. Optimizar la seguridad en la red lo más posible.

## 3.2. Evaluar y seleccionar tecnología

### 3.2.1. Aspectos Tecnológicos a Considerar

La cantidad de empleados en la tienda es de seis; el gerente propietario, un administrador, un facturador, dos vendedores, y uno de almacén.

Con respecto al tipo de red a utilizar en la tienda según su clasificación se usara: Utilizaremos una red de área local (LAN) con topología de estrella, ya que todas las estaciones estarán conectadas individualmente a un servidor a través de un cable de red punto a punto, los cuales se comportan uno transmisor y otro como receptor.

El comportamiento del nodo central (Switch) sería como dispositivo de computo de tramas: debido a que la trama entrante se almacenara en el nodo y se retransmitirá sobre un enlace de salida a la estación destino (Servidor).

La red contara con un servicio independiente de Internet únicamente utilizado por el propietario y administrador. El protocolo para la red a utilizar será TCP/IP, con IP estáticos.

### 3.2.2. Funciones y características de la tecnología

#### 3.2.2.1. Características geográficas del lugar.

Las dimensiones de la tienda son de 30 mts de ancho, 30 mts de largo y 4 mts de alto.
El lugar se compone de dos oficinas: la del propietario y la del administrador, un área de bodega y un área mayor donde está ubicada venta y facturación.

#### 3.2.2.2. Tecnología a utilizar

El servidor estará ubicado donde está el administrador y posee las siguientes características: CPU marca Dell con 2 GB de memoria RAM, un disco duro de 160 GB, procesador Intel de 2.8 Ghz, AT/AT compatible y sistema operativo Windows XP. Las especificaciones de las estaciones a configurarse como cliente tendrán: 512 de RAM, disco duro de 40 GB, procesador Intel de 1.6 GHz y sistema operativo Windows XP.

Para la red interna utilizaremos el tipo de direccionamiento IP Clase C 220.10.10.0. El medio de transmisión de la red será cable UTP categoría 5, con la capacidad de sostener comunicaciones a 100 Mbps. Todas las conexiones de cableado se rigen por el estándar de la norma EIA/TIA 568B con el orden de colores, blanco_naranja–naranja, blanco_verde–azul, blanco_azul–verde y blanco_caje–café. **Ver Anexo IV**. Los conectores RJ 45 para el cableado soportan esta norma.

Los medios de comunicación propuestos a la tienda para la red serán de un switch para la red de Internet y la red de sistema interno. A continuación se detallan las especificaciones del medio (switch).

**SWITCH DE 16 DE PUERTOS**

- **Descripción:** 3Com Baseline Switch 2816
- **Características:** Control de flujo, capacidad dúplex, conmutación Layer 2, negociación automática.
- **Tecnología de conectividad:** Cableado
- **Protocolo de interconexión de datos:** Ethernet, Fast Ethernet, Gigabit Ethernet.
- **Normas:** IEEE 802.33, IEEE 802.3U, 802.1D, IEEE 802.3ab, IEEE 802.1p, IEEE 802.3x.
- **Puertos:** 16 x Ethernet 10Base-T, Ethernet 100Base_TX, Ethernet 1000Base-T.
- **Velocidad de Transferencia de datos:** 1Gbps.
- **Modo de Comunicación:** Semiduplex, dúplex pleno.
- **Protocolo de Comunicación:** Ethernet.
- **Tamaño de tabla de dirección MAC:** 32K de entradas.
- **Normas:** Certificado FCC Clase A, CSA, EN 60950, EN55022, ICES-003, UL 1950, VCCI Class A ITE, IEC 60950, EN55024.
- **Interfaces:** 16 x red – Ethernet 10Base-T/100Base-TX/1000Base-T-RJ-45.

Las tarjetas a utilizar en las maquinas serán 3Com.

**Trafico en la Red**

Las cinco estaciones estarán distribuidas de la siguiente forma:

- Una en la oficina del propietario.
- Una en administración.
- Una en almacén.
- Una en facturación.
- Una en ventas.

### 3.3. Desarrollar un plan de interconectividad

Conforme al diseño de red que se está proponiendo consta de solo cinco estaciones no sería necesario plantearnos un mecanismo de interconexión, ya que todo estará comprendido en un mismo grupo (área) de trabajo.

### 3.4. Considerar Administración y seguridad de la red

**Seguridad de la red**

**Seguridad física.** Englobaremos dentro de esta categoría a todos los asuntos relacionados con la salvaguarda de los soportes físicos de la información. En este nivel estarán, entre otras, las medidas contra incendios y sobrecargas eléctricas, las políticas de backup, etc. También se suelen tener en cuenta dentro de este punto aspectos relacionados con la restricción de acceso físico a las computadoras únicamente a personas autorizadas.

- ✓ Cubrir los cables de red para evitar exposición y deterioro de los mismos.
- ✓ Colocar el panel de conexión central en un área de acceso restringido.
- ✓ Colocar extinguidotes en un lugar estratégico.
- ✓ Disponer de estabilizadores de corriente en cada estación de trabajo.

**Seguridad de la información.** En esta parte prestaremos atención a la preservación de la información frente a observadores no autorizados. Para ello podemos emplear tanto criptografía simétrica como asimétrica, estando la primera únicamente indicada en sistemas aislados, ya que si la empleáramos en redes, al tener que transmitir la clave por el canal de comunicación, estaremos asumiendo un riesgo excesivo.

- ✓ Realizar respaldos de la información esporádicamente en medios de almacenamiento externo.

**Seguridad del canal de comunicación.** Los canales de comunicación rara vez se consideran seguros. Debido a que en la mayoría de los casos escapan a nuestro control, ya que pertenecen a terceros, resulta imposible asegurarse totalmente de que no están siendo escuchados o intervenidos.

**Problemas de autenticación.** Debido a los problemas del canal de comunicación, es necesario asegurarse de que la información que recibimos en la computadora viene de quien realmente creemos que viene. Para esto se suele emplear criptografía asimétrica en conjunción con funciones resumen.

**Problemas de suplantación.** En las redes tenemos el problema añadido de que cualquier usuario autorizado puede acceder al sistema desde fuera, por lo que hemos de confiar en sistemas fiables para garantizar que los usuarios no están siendo suplantados por intrusos. Para ello emplearemos mecanismos basados en password para conseguir la autenticidad del usuario.

**Políticas de Uso de la Red de Comunicación**

**Niveles de trabajo en la red:**

- ✓ Confidencialidad
- ✓ Integridad
- ✓ Autenticidad
- ✓ No repudio
- ✓ Disponibilidad de los recursos y de la informática
- ✓ Consistencia
- ✓ Control de acceso

**Confidencialidad:** Consiste en proteger la informática contra la lectura no autorizada explícitamente. Incluye no solo la protección de la informática en su totalidad, sino también las piezas individuales que pueden ser utilizadas para inferir otros elementos de información confidencial.

En nuestro caso se usara la autentificación de usuario para así evitar cambios no autorizados a la información de la tienda. Los niveles de usuario serán otorgados según las especificaciones del propietario y el área de trabajo en que se encuentra en la tienda.

**Integridad:** Es necesario proteger la información contra la modificación sin permiso del propietario. La información al ser protegida incluye no solo la que esta almacenada directamente en el sistema sino que también se deben considerar elementos menos obvios como respaldos, documentación, registros de contabilidad del sistema, transito en la red, etc.
Esto comprende cualquier tipo de modificación:

- ✓ Causadas por errores de hardware y/o software.
- ✓ Causadas de formar intencional.
- ✓ Causadas de forma accidental.

Cuando se trabaja con una red, se debe comprobar que los datos no fueron modificados durante su transferencia.

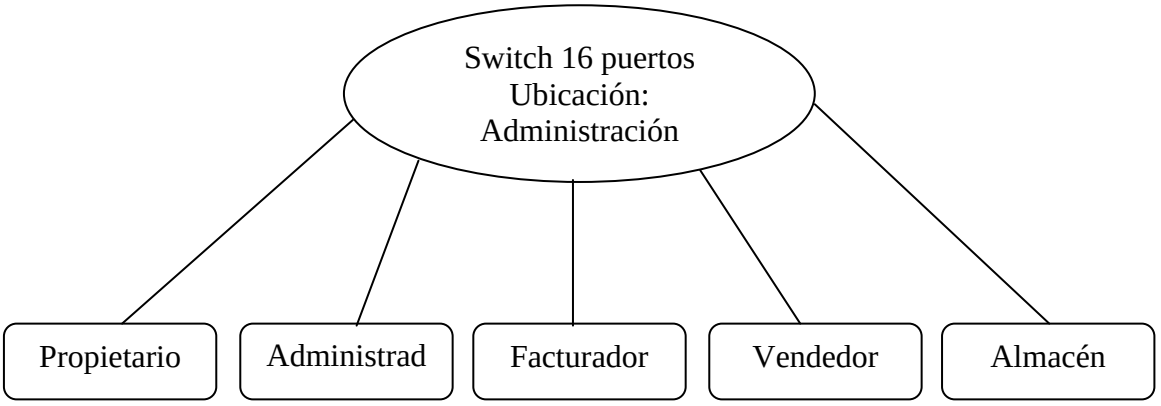
**Autenticidad:** En cuanto a telecomunicaciones se refiere, la autenticidad garantiza que dice ser “X” es realmente “X”. Es decir, se debe implementar mecanismos para verificar quien esta enviando la información.

**No Repudio:** Ni el origen ni el destino en un mensaje deben poder negar la transmisión. Quien envía el mensaje puede probar que, en efecto, el mensaje fue enviado y viceversa.

**Consistencia:** Se trata de asegurar que el sistema siempre se comporte de la forma esperada, de tal manera que los usuarios no encuentren variantes inesperadas.

**Control de Acceso:** Consiste en controlar quien utiliza el sistema o cualquiera de los recursos que ofrece y como lo hace, mediante autenticación de usuario.

**Diseño Lógico de la Red**



# 4. Diseño Físico

## 4.1. Diseño del Cableado

Estará basada en el modo de distribución centralizado, lo que quiere decir que todas las estaciones estarán a un punto central (Switch), el cual estará conectado directamente al servidor

Para definir el sistema de cableado por el cual se regirá nuestro proyecto, consideraremos las normas que establece el sistema de cableado centralizado, específicamente adoptaremos la norma 568-A la cual se fundamenta en que permite diseñar e instalar el cableado de telecomunicaciones contando con poca información acerca de los productos de telecomunicaciones que posteriormente se instalarán. Como medio físico se utilizará el cable UTP nivel 5, ya que este permite mayor rapidez para el manejo de información y es el mas utilizado y recomendado en el mercado. Este medio físico tendrá una longitud máxima de 100 mts, tal y como lo establecen las normas del C.E.

## 4.2. Valoraciones medio ambientales

Actualmente la tienda cuenta con un sistema de ventilación (Aire Acondicionado) ubicado en un punto estratégico de tal manera que todas las áreas donde se vaya a colocar una estación estén debidamente ventiladas.

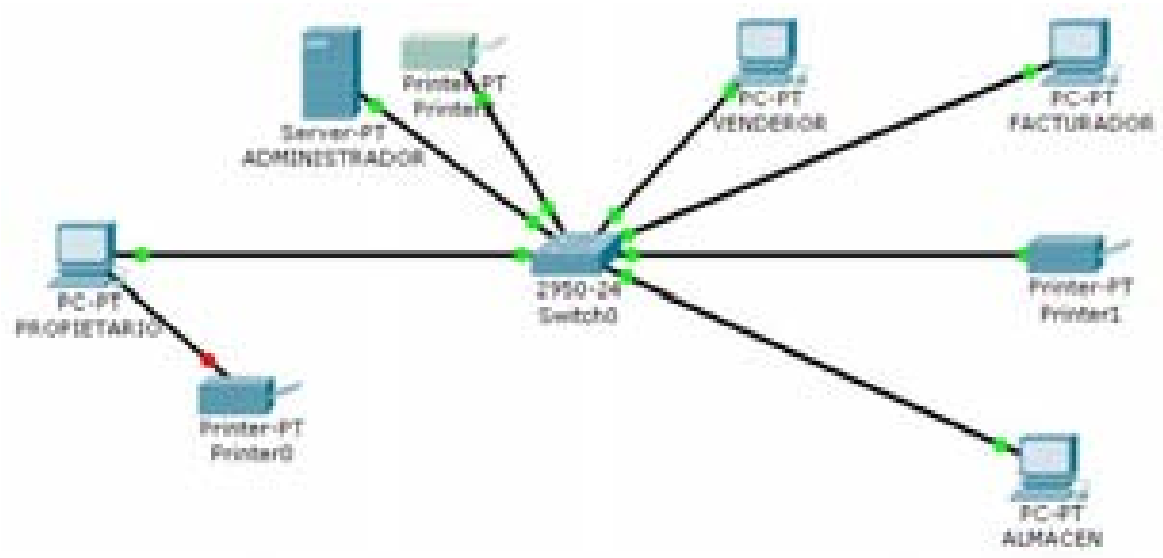
El área de facturación y venta abarca 17 metros donde se encuentra debidamente acomodados los productos que ofrece la tienda. El tamaño de la oficina del propietario y el administrador son de 3 metros y almacén abarca 7 metros.

### 4.3. Ubicación de Equipos

Las distancias que existirán entre las estaciones de trabajo y el switch son las que especificaremos a continuación, utilizaremos metros como unidad de medición para estas distancias:

| Estación de Trabajo | Switch |
|---|---|
| Propietario | 5 mts |
| Administrador | 2.5 mts |
| Facturador | 10 mts |
| Vendedor | 12 mts |
| Almacén | 20 mts |

**Diseño Físico de la Red.**



# V. DIRECCIONAMIENTO Y RUTEO

## DIRECCIONAMIENTO Y RUTEO

## DIRECCIONES IP, PUERTA DE ENLACE Y DNS

**IP PUBLICA:** Es la IP que nos asignara el proveedor de Internet, con el fin de identificarnos. Se otorgara una IP Dinámica que es lo más habitual en este tipo de servicio.

Un ejemplo de la IP PUBLICA que se le podría asignar a la tienda seria la siguiente: 192.168.0.0. La cual es una dirección de clase “C”.

**PUERTA DE ENLACE:** Un nuestro proyecto no se aplica este mecanismo de direccionamiento, debido que esto aplica para identificar a los Router que utilicemos en la red. Pero ejemplo de esta podría ser: 192.168.1.1.

**IP PRIVADAS:** Son las IP que aplicaremos a los ordenadores en la red interna de la Tienda, debido al tamaño de la red se aplicare u direccionamiento de clase “C”. A continuación mencionamos las especificaciones de cada estación de trabajo:

| Ubicación | IP | Nombre Equipo | Grupo de Trabajo | MAC |
|---|---|---|---|---|
| Gerencia | 200.10.10.1 | Propietario | TD | 0000.0C36.7461 |
| Facturación | 200.10.10.2 | Facturador | TD | 0001.63D5.55BE |
| Ventas | 200.10.10.3 | Vendedor | TD | 0060.7097.4582 |
| Almacén | 200.10.10.4 | Almacén | TD | 0001.C762.B706 |
| Administración | 200.10.10.5 | Administrador | TD | 0060.5C96.7981 |

Las mascara de red para todas las estaciones de trabajo es la siguiente: 255.255.255.0.

# VI. PROPUESTA DE INVERSION

**Propuesta de Inversión del Proyecto**

El sistema de facturación y control de inventario esta dirigido a las tareas realizadas por los usuarios en cada área de trabajo. En el presente trabajo se propondrá la incorporación de 5 PC una en cada estación de trabajo, tres impresoras. Todo esto estará conectado mediante una red que estará montada en un Switch de 16 puertos el cual estará conectada a un servidor que se comportara como servidor. La cantidad de cableado y demás elementos de la red se detallan a continuación:

| **Estación de Trabajo** | **Switch** |
|---|---|
| Propietario | 5.5 mts |
| Administrador | 2.5 mts |
| Facturador | 10 mts |
| Vendedor | 12 mts |
| Almacén | 20 mts |
| **Total** | **50 mts** |

Debido a que actualmente en la tienda no se cuenta con ninguna estación para ser montada en la red a continuación se presenta la propuesta de todos los equipos que se van a implementar para poder montar la red propuesta. Con respecto al servicio de Internet se contratara el servicio a una compañía externa, el promedio mensual del costo de este servicio en el mercado es de US$ 50.00. La contratación de este servicio dependerá del propietario. Propuesta de computadoras y accesorios a utilizar.
(Ver Anexo IV):

| Cantidad | Descripción | Área | Importe US$ |
|---|---|---|---|
| 1 | PC y Accesorios. Propuesta 1 | Administración. Servidor | 480.00 |
| 1 | PC y Accesorios. Propuesta 1 | Propietario | 480.00 |
| 1 | PC y Accesorios. Propuesta 2 | Facturación | 300.00 |
| 1 | PC y Accesorios. Propuesta 2 | Ventas | 300.00 |
| 1 | PC y Accesorios. Propuesta 2 | Almacén | 300.00 |
| 3 | Impresora HP 1230N Láser | Facturación, Administración y Propietario | 150.00 |
| 1 | Switch 16 puertos | | 800.00 |
| 50 mts | Cable UTP Cat. 5 | | 20.00 |
| 20 | Clavijas RJ45 | | 6.00 |
| 20 | Canaletas de 2 mts | | 60.00 |
| | | **SUBTOTAL** | **2,896.00** |
| | | **I.V.A** | **434.40** |
| | | **TOTAL** | **3,330.40** |

# CONCLUSIONES

En el presente proyecto se tomo como fase de inicio realizar un Enfoque Sistémico para estudiar la problemática que existe en todos las actividades que se realizan en la tienda DENMAR, con lo cual se propuso las posibles soluciones para todos los problemas que se encontraron en el negocio.

Se elaboro un Diagrama de Casos de Uso para proponer de forma general la funcionalidad del Prototipo de Sistema de Facturacion y Control de Inventario que se propone para la tienda.

Se construyo el Diagrama de Clases y Modelo de Datos para plantear la estructura y lo métodos que contiene la base de datos, la cual esta elaborada en el gestor de almacenamiento SQL SERVER 2000.

Se proporcionó las posibles pantallas que contendrá el Prototipo de Sistema de Facturación y Control de Inventario para la tienda DENMAR mediante la plataforma de programación Netbeans 6.0.

Por ultimo se Diseño un Diagrama Físico de Red, por medio del cual se propone la forma de cómo estarán distribuidos todos los equipos que van a utilizar el prototipo de Sistema de Facturacion y Control de Inventario.

## RECOMENDACIONES

- Se aconseja al propietario de la tienda poner en práctica el manual de funciones propuesto para los empleados de la tienda para llevar de forma eficiente el control de las actividades de la misma.

- Se recomienda al propietario agotar todos los recursos posibles de financiamiento para implementar el Protitopo de Sistema de Facturación y Control de Inventario, ya que este le traerá mayores beneficios en un futuro inmediato.

# ANEXOS

# ANEXO I

## ENTREVISTA

Con la dueña

Gerente Propietaria

1. ¿Qué función ocupa usted en esta microempresa?

2. ¿Cuántos empleados tiene la microempresa y que funciones desempeñan?

3. ¿Cómo opera actualmente la tienda “DENMAR”?

4. ¿Quién Maneja el comportamiento de compra y venta?

5. ¿Posee la entidad un departamento de recursos humanos?

6. ¿Cuál es el método de facturación en el momento de que el cliente realiza una compra?

7. ¿Le interesa a usted que le elaboremos un sistema de facturación y control de inventario para esta entidad?

8. ¿Qué contiene su inventario?, es decir, nos referimos al tipo de inventario que mantiene regularmente en funcionamiento.

9. ¿Cuáles son las unidades de medida que emplea para el control de los productos?

10. ¿Qué mecanismo usa para el control de la entrada o salida de los productos?

11. Cambiando completamente de tema, ¿quiénes conforman la estructura organizacional de su empresa y qué tareas se les tienen asignadas?

12. Ahora, referente a los productos, ¿hay diferentes tipos de ventas?, si las hay, ¿cuáles son?

13. ¿Usted cobra el IVA. por la venta de cualquiera de sus productos?

14. ¿Qué tipos de proveedores suministran los productos que comercializan y como paga?

15. Regresando a las ventas, ¿le aplican retenciones sobre el monto de la compra de sus productos?

16. Con respecto al uso del Software que le desarrollaremos, ¿tiene alguna experiencia en la utilización de software para computadoras?, ¿estudió algún curso de computación?

17. ¿Cuántas personas utilizarán la aplicación?, ¿quiénes son?

# ANEXO II

## CODIGO BASE DE DATOS

## CREACION DE BASE DE DATOS Y TABLAS

```
/* CREA LA BASE DE DATOS TIENDA */
CREATE DATABASE TIENDA
GO

/* MUESTRA LA INFORMACION DE LA BASE DE DATOS CREADA */
SP_HELPDB TIENDA
GO

USE TIENDA
GO

/* CREANDO LAS TABLAS */

/*DETALLE_CATEGORIA*/
-------------------------------------------------------------------------
CREATE TABLE DETALLE_CATEGORIA
(Cod_Categoria varchar(15) CONSTRAINT PK_CATEGORIA Primary Key,
Categoria varchar(50) not null)
GO

SP_HELP DETALLE_CATEGORIA
GO
-------------------------------------------------------------------------

/*DETALLE_PRODUCTO*/
-------------------------------------------------------------------------
CREATE TABLE DETALLE_PRODUCTO
(Cod_Subcategoria varchar(15) CONSTRAINT PK_SUBCATEGORIA Primary Key,
Cod_Categoria varchar(15) CONSTRAINT FK_CODCATEGORIA REFERENCES
DETALLE_CATEGORIA(Cod_Categoria) not null,
SubCategoria varchar(50) not null,
Tamaño varchar(20) not null,
Marca varchar(20) not null)
GO

SP_HELP DETALLE_PRODUCTO
GO
-------------------------------------------------------------------------
drop table PRODUCTO

/*PRODUCTO*/
-------------------------------------------------------------------------
CREATE TABLE PRODUCTO
(Cod_Producto varchar(15) CONSTRAINT PK_PRODUCTO Primary Key,
Cod_Categoria varchar(15) CONSTRAINT FK_CATEGORIA REFERENCES
DETALLE_CATEGORIA(Cod_Categoria) not null,
Cod_Subcategoria varchar(15) CONSTRAINT FK_SUBCATEGORIA REFERENCES
DETALLE_PRODUCTO(Cod_Subcategoria) not null,
Precio_Compra Float not null,
Precio_Venta Float not null,
```

```
Existencia Integer CONSTRAINT CK_EXIS CHECK(Existencia>0))
GO

SP_HELP PRODUCTO
GO
-----------------------------------------------------------------------

/*PROVEEDOR*/
-----------------------------------------------------------------------
CREATE TABLE PROVEEDOR
(Cod_Proveedor varchar(15) CONSTRAINT PK_PROVEEDOR Primary Key,
Nombre_Proveedor varchar(30) not null,
Direccion varchar(50) not null,
Telefono varchar(10) not null)
GO

SP_HELP PROVEEDOR
GO
-----------------------------------------------------------------------

/*FACTURAR_VENTA*/
-----------------------------------------------------------------------
CREATE TABLE FACTURAR_VENTA
(Cod_Factura_Venta varchar(15) CONSTRAINT PK_FACTURARVENTA Primary Key,
Fecha_Venta datetime not null,
Descripcion varchar(100) not null,
Nombre_Cliente varchar(50) not null,
Subtotal float CONSTRAINT CK_SUBTOTAL DEFAULT 0.00 not null,
IVA float CONSTRAINT CK_IVA DEFAULT 0.00 not null,
Total float CONSTRAINT CK_TOTAL DEFAULT 0.00 not null,
Pago float CONSTRAINT CK_PAGO DEFAULT 0.00 not null,
Vuelto float CONSTRAINT CK_VUELTO DEFAULT 0.00 not null)
GO

SP_HELP FACTURAR_VENTA
GO
-----------------------------------------------------------------------

/*DETALLE_FACTURA_VENTA*/
-----------------------------------------------------------------------
CREATE TABLE DETALLE_FACTURA_VENTA
(Id_Detalle_Venta Integer Identity(1,1) CONSTRAINT PK_DETALLEVENTA Primary Key,
Cod_Factura_Venta varchar(15) CONSTRAINT FK_FACTURA_VENTA REFERENCES
FACTURAR_VENTA(Cod_Factura_Venta) not null,
Cod_Producto varchar(15) not null,
Cantidad integer not null,
Precio float not null,
Subtotal float CONSTRAINT CK_SUBTOTAL1 DEFAULT 0.00 not null,
Descuento float CONSTRAINT CK_DESCUENTO DEFAULT 0.00 not null,
Total float CONSTRAINT CK_TOTAL1 DEFAULT 0.00 not null)
GO

SP_HELP DETALLE_FACTURA_VENTA
GO
-----------------------------------------------------------------------
```

```
/*FACTURAR_COMPRA*/
-------------------------------------------------------------------------
CREATE TABLE FACTURAR_COMPRA
(Cod_Factura_Compra varchar(15) CONSTRAINT PK_FACTURARCOMPRA Primary Key,
Cod_Proveedor varchar(15) CONSTRAINT FK_PROVEEDOR REFERENCES
PROVEEDOR(Cod_Proveedor) not null,
Fecha_Compra datetime not null,
Total float CONSTRAINT CK_TOTAL2 DEFAULT 0.00 not null)
GO

SP_HELP FACTURAR_COMPRA
GO
-------------------------------------------------------------------------

/*DETALLE_FACTURA_COMPRA*/
-------------------------------------------------------------------------
CREATE TABLE DETALLE_FACTURA_COMPRA
(Id_Detalle_Compra Integer Identity(1,1) CONSTRAINT PK_DETALLECOMPRA Primary Key,
Cod_Factura_Compra varchar(15) CONSTRAINT FK_FACTURA_COMPRA REFERENCES
FACTURAR_COMPRA(Cod_Factura_Compra) not null,
Cod_Producto varchar(15) not null,
Producto varchar(100) not null,
Cantidad integer not null,
Precio float not null,
Subtotal float CONSTRAINT CK_SUBTOTAL2 DEFAULT 0.00 not null)
GO

SP_HELP DETALLE_FACTURA_COMPRA
GO
-------------------------------------------------------------------------

/*ARQUEO*/
-------------------------------------------------------------------------
CREATE TABLE ARQUEO
(Cod_Arqueo varchar(15) CONSTRAINT PK_CODARQUEO Primary Key,
Fecha_Arqueo datetime not null,
Total_Facturas_Dia float CONSTRAINT CK_TOTAL_FACTURAS_DIA DEFAULT 0.00 not null,
Monto_Caja float CONSTRAINT CK_MONTO_CAJA DEFAULT 0.00 not null,
Diferencia float CONSTRAINT CK_DIFERENCIA DEFAULT 0.00 not null)
GO

SP_HELP ARQUEO
GO
-------------------------------------------------------------------------


--------------------------- FOREIGN KEY FALTANTES ------------------------
USE TIENDA
GO

ALTER TABLE DETALLE_FACTURA_COMPRA
ADD CONSTRAINT FK_COD_PRODUCTO_COMPRA
FOREIGN KEY (Cod_Producto)
REFERENCES PRODUCTO(cod_Producto)
GO
```

```
ALTER TABLE DETALLE_FACTURA_VENTA
ADD CONSTRAINT FK_COD_PRODUCTO_VENTA
FOREIGN KEY (Cod_Producto)
REFERENCES PRODUCTO(cod_Producto)
GO
```

---

## CREACION DE TRIGGERS

```
USE TIENDA
GO

SELECT * FROM PRODUCTO
SELECT * FROM DETALLE_FACTURA_VENTA
SP_HELP DETALLE_FACTURA_VENTA

                    /* CONTROL DE INVENTARIO */

/* SE REALIZA UNA VENTA */
```

---

```
CREATE TRIGGER VENTA
ON DETALLE_FACTURA_VENTA
FOR INSERT
AS

        UPDATE PRODUCTO
        SET EXISTENCIA = EXISTENCIA - INSERTED.CANTIDAD
        FROM PRODUCTO INNER JOIN INSERTED
        ON PRODUCTO.Cod_Producto = INSERTED.Cod_Producto

GO
```

---

```
/* SE ELIMINA UNA ENTA */
```

---

```
CREATE TRIGGER BORRARVENTA
ON DETALLE_FACTURA_VENTA
FOR DELETE
AS
        UPDATE PRODUCTO
        SET EXISTENCIA = EXISTENCIA + DELETED.CANTIDAD
        FROM PRODUCTO INNER JOIN DELETED
        ON PRODUCTO.Cod_Producto = DELETED.Cod_Producto
GO
```

---

```
/* SE REALIZA UNA COMPRA */
```

---

```
CREATE TRIGGER COMPRA
ON DETALLE_FACTURA_COMPRA
FOR INSERT
AS
        UPDATE PRODUCTO
        SET EXISTENCIA = EXISTENCIA + INSERTED.CANTIDAD
```

```
        FROM PRODUCTO INNER JOIN INSERTED
        ON PRODUCTO.Cod_Producto = INSERTED.Cod_Producto
GO
-------------------------------------------------------------------------------


/* SE ELIMINA UNA COMPRA */
-------------------------------------------------------------------------------
CREATE TRIGGER BORRARCOMPRA
ON DETALLE_FACTURA_COMPRA
FOR DELETE
AS
        UPDATE PRODUCTO
        SET EXISTENCIA = EXISTENCIA - DELETED.CANTIDAD
        FROM PRODUCTO INNER JOIN DELETED
        ON PRODUCTO.Cod_Producto = DELETED.Cod_Producto
GO
-------------------------------------------------------------------------------
```

## CREACION DE LOGIN

```
USE TIENDA
GO

SELECT * FROM PROVEEDOR
GO

EXEC sp_addlogin 'ajqm','1111','TIENDA'
go
EXEC sp_grantdbaccess 'ajqm'
go
--Agrega o retira usuarios y/o roles
EXEC sp_addrolemember 'db_accessadmin', 'ajqm'
go

SP_HELPUSER ajqm
GO
```

# ANEXO III

## CODIGO PROTOTIPO SISTEMA
## CLASES

### CONECCION1

```
package CLASES;

import java.beans.Statement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;


public class CONECCION {
public static Connection conexion = null;
public Statement consulta = null;
public ResultSet result = null;
public PreparedStatement pConsulta = null;

    public CONECCION() {
        try
        {
          String database="TIENDA";
          String login = "sa";
          String passwd = "sa";
          Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver"); /*Cargar driver 2000*/
          conexion =
DriverManager.getConnection("jdbc:microsoft:sqlserver://localhost:1433;database = "+
database,login,passwd);
          if(conexion.isClosed())  /*prueba si la conexion se llevo a cabo*/
          {
              System.out.println("Error no se pudo conectar");
          }
          else
          {
              System.out.println("Conexion Satisfactoria");
          }

        } catch(ClassNotFoundException ex)
        {
        }
          catch(SQLException ex2)
        {
            System.out.print(ex2.getMessage());
        }

    }

public static void main(String...args) throws SQLException
    {
        CONECCION con = new CONECCION();
```

```
        if(con == null)
            System.out.println("Error no se pudo conectar");
    }
}
```

## CONECCION2

```
package CLASES;

import java.beans.Statement;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class CONECCION2 {
 private static String usuario;
    public  static Connection conexion = null;
    public static Statement consulta = null;
    public static PreparedStatement pConsulta = null;
    public static ResultSet registros = null;
    public static CallableStatement pProcedimiento = null;

    public static String getUsuario() {
        return usuario;
    }
    public CONECCION2() {
    }

    public CONECCION2(String usuario, String clave) {
        try
        {
            CONECCION2.usuario = usuario;
            String database = "TIENDA";
            Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver"); /*Cargar driver 2000*/
            conexion =
DriverManager.getConnection("jdbc:microsoft:sqlserver://localhost:1433;database = "+
database,usuario,clave);
            if(conexion != null)
                System.out.println("Conexion Satisfactoria");
        }   catch(SQLException ex2) {
            System.out.print(ex2.getMessage());
            JOptionPane.showMessageDialog(null, "El registro de usuario ha fallado");
        }
        catch(ClassNotFoundException ex){
JOptionPane.showMessageDialog(null,ex.getMessage()); }
    }
}
```

## VALIDACION

```
package CLASES;

import javax.swing.JOptionPane;

public class Validacion {
private String texto;
private char caracter;

   public Validacion(String texto, char digito) {
      this.texto = texto;
      this.caracter = digito;
   }
public boolean validarFloat()
   {
      if(!(Character.isDigit(caracter)) && !(caracter == '.'))
      {
         return false;
      }
      else if(caracter == '.')
      {
         if(texto.length() == 0)
         {
           return false;
         }
         else
         {
            for(int i=1;i<texto.length();i++)
            {
              if(texto.charAt(i)=='.')
              {
                return false;
              }
            }
         }
      }
      return true;
   }
   public boolean validarEntero()
   {
      if(!(Character.isDigit(caracter)))
        return false;
      return true;
   }
   public boolean validarEntero(int maximo)
   {
      if(!(Character.isDigit(caracter)))
      {
         return false;
      }
      if(texto.length()>0)
      {
           int valor = Integer.parseInt(texto+caracter);
           if(valor > maximo)
```

```
            {
            JOptionPane.showMessageDialog(null, "El numero debe ser menor o igual a "+maximo);
            return false;
            }
        }
        return true;
    }
}
```

## SQLEXCEPTION

```
package CLASES;
import java.sql.SQLException;

public class SQLEXCEPTION extends SQLException {
 private String message;
    public SQLEXCEPTION(String message) {
        super(message);
        this.message = message;
    }
}
```

## VENTA

```
package CLASES;

import java.beans.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class VENTA {
private String cod_factura, fecha, descripcion, cliente;
private Double subtotal, iva, total, pago, vuelto;
private ResultSet result = null;
private Statement consulta = null;

    public VENTA() {
    }

    public VENTA(String cod_factura, String fecha, String descripcion, String cliente, Double
subtotal, Double iva, Double total, Double pago, Double vuelto) {
        this.cod_factura = cod_factura;
        this.fecha = fecha;
        this.descripcion = descripcion;
        this.cliente = cliente;
        this.subtotal = subtotal;
        this.iva = iva;
        this.total = total;
        this.pago = pago;
        this.vuelto = vuelto;
    }
    public String getCliente() {
        return cliente;
    }

    public void setCliente(String cliente) {
        this.cliente = cliente;
```

```
}

public String getCod_factura() {
    return cod_factura;
}

public void setCod_factura(String cod_factura) {
    this.cod_factura = cod_factura;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

public String getFecha() {
    return fecha;
}

public void setFecha(String fecha) {
    this.fecha = fecha;
}
public Double getIva() {
    return iva;
}
public void setIva(Double iva) {
    this.iva = iva;
}
public Double getPago() {
    return pago;
}
public void setPago(Double pago) {
    this.pago = pago;
}
public Double getSubtotal() {
    return subtotal;
}
public void setSubtotal(Double subtotal) {
    this.subtotal = subtotal;
}
public Double getTotal() {
    return total;
}
public void setTotal(Double total) {
    this.total = total;
}
public Double getVuelto() {
    return vuelto;
}
public void setVuelto(Double vuelto) {
    this.vuelto = vuelto;
}
```

```
public void insertarDatos(String codfactura, String fecha, String descripcion, String cliente,
                    double subtotal, double iva, double total, double pago, double vuelto) throws
SQLException {
  String sql = "INSERT INTO FACTURAR_VENTA(Cod_Factura_Venta, Fecha_Venta,
Descripcion, Nombre_Cliente," +
           "Subtotal, IVA, Total, Pago, Vuelto) VALUES(?,?,?,?,?,?,?,?,?)";

      PreparedStatement consultaP = CONECCION.conexion.prepareStatement(sql);

      consultaP.setString(1, codfactura);
      consultaP.setString(2, fecha);
      consultaP.setString(3, descripcion);
      consultaP.setString(4, cliente);
      consultaP.setDouble(5, subtotal);
      consultaP.setDouble(6, iva);
      consultaP.setDouble(7, total);
      consultaP.setDouble(8, pago);
      consultaP.setDouble(9, vuelto);
      consultaP.executeUpdate();
   }
}
```

## DETALLE VENTA

```
package CLASES;

public class DETALLEVENTA {
private String cod_factura, cod_producto;
private int cantidad;
private Double precio, subtotal, descuento, total;

   public DETALLEVENTA() {
   }

   public DETALLEVENTA(String cod_factura, String cod_producto, int cantidad, Double precio,
Double subtotal, Double descuento, Double total) {
      this.cod_factura = cod_factura;
      this.cod_producto = cod_producto;
      this.cantidad = cantidad;
      this.precio = precio;
      this.subtotal = subtotal;
      this.descuento = descuento;
      this.total = total;
   }

   public int getCantidad() {
      return cantidad;
   }

   public void setCantidad(int cantidad) {
      this.cantidad = cantidad;
   }

   public String getCod_factura() {
      return cod_factura;
   }
```

```
    public void setCod_factura(String cod_factura) {
        this.cod_factura = cod_factura;
    }

    public String getCod_producto() {
        return cod_producto;
    }

    public void setCod_producto(String cod_producto) {
        this.cod_producto = cod_producto;
    }

    public Double getDescuento() {
        return descuento;
    }

    public void setDescuento(Double descuento) {
        this.descuento = descuento;
    }

    public Double getPrecio() {
        return precio;
    }

    public void setPrecio(Double precio) {
        this.precio = precio;
    }

    public Double getSubtotal() {
        return subtotal;
    }
    public void setSubtotal(Double subtotal) {
        this.subtotal = subtotal;
    }
    public Double getTotal() {
        return total;
    }
    public void setTotal(Double total) {
        this.total = total;
    }
}
```

## COMPRA

```
package CLASES;

public class COMPRA {
private String cod_factura, cod_proveedor,fecha;
private double total;

    public COMPRA() {
    }

  public COMPRA(String cod_factura, String cod_proveedor, String fecha, double total) {
        this.cod_factura = cod_factura;
        this.cod_proveedor = cod_proveedor;
```

```
        this.fecha = fecha;
        this.total = total;
    }

    public String getCod_factura() {
        return cod_factura;
    }

    public void setCod_factura(String cod_factura) {
        this.cod_factura = cod_factura;
    }

    public String getCod_proveedor() {
        return cod_proveedor;
    }

    public void setCod_proveedor(String cod_proveedor) {
        this.cod_proveedor = cod_proveedor;
    }

    public String getFecha() {
        return fecha;
    }

    public void setFecha(String fecha) {
        this.fecha = fecha;
    }



    public double getTotal() {
        return total;
    }
    public void setTotal(double total) {
        this.total = total;
    }
}
```

## DETALLE COMPRA

```
package CLASES;

public class DETALLECOMPRA {
private String cod_factura, cod_producto, producto;
private int cantidad;
private Double precio,total;

    public DETALLECOMPRA() {
    }

    public DETALLECOMPRA(String cod_factura, String cod_producto, String producto, int
cantidad, Double precio, Double total) {
        this.cod_factura = cod_factura;
        this.cod_producto = cod_producto;
        this.producto = producto;
        this.cantidad = cantidad;
```

```
        this.precio = precio;
        this.total = total;
    }
    public int getCantidad() {
        return cantidad;
    }
    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }
    public String getCod_factura() {
        return cod_factura;
    }
    public void setCod_factura(String cod_factura) {
        this.cod_factura = cod_factura;
    }
    public String getCod_producto() {
        return cod_producto;
    }
    public void setCod_producto(String cod_producto) {
        this.cod_producto = cod_producto;
    }

    public Double getPrecio() {
        return precio;
    }

    public void setPrecio(Double precio) {
        this.precio = precio;
    }
    public String getProducto() {
        return producto;
    }
.

    public void setProducto(String producto) {
        this.producto = producto;
    }
    public Double getTotal() {
        return total;
    }
    public void setTotal(Double total) {
        this.total = total;
    }
}
```

## PRODUCTO

```
package CLASES;

public class PRODUCTO {
private String cod_producto, cod_categoria,cod_subcategoria;
private double precio_compra, precio_venta;
private int existencia;

    public PRODUCTO() {
    }
```

```
    public PRODUCTO(String cod_producto, String cod_categoria, String cod_subcategoria,
double precio_compra, double precio_venta, int existencia) {
        this.cod_producto = cod_producto;
        this.cod_categoria = cod_categoria;
        this.cod_subcategoria = cod_subcategoria;
        this.precio_compra = precio_compra;
        this.precio_venta = precio_venta;
        this.existencia = existencia;
    }

    public String getCod_categoria() {
        return cod_categoria;
    }

    public void setCod_categoria(String cod_categoria) {
        this.cod_categoria = cod_categoria;
    }


    public String getCod_producto() {
        return cod_producto;
    }
    public void setCod_producto(String cod_producto) {
        this.cod_producto = cod_producto;
    }
    public String getCod_subcategoria() {
        return cod_subcategoria;
    }
    public void setCod_subcategoria(String cod_subcategoria) {
        this.cod_subcategoria = cod_subcategoria;
    }
    public int getExistencia() {
        return existencia;
    }
    public void setExistencia(int existencia) {
        this.existencia = existencia;
    }
    public double getPrecio_compra() {
        return precio_compra;
    }
    public void setPrecio_compra(double precio_compra) {
        this.precio_compra = precio_compra;
    }
    public double getPrecio_venta() {
        return precio_venta;
    }
    public void setPrecio_venta(double precio_venta) {
        this.precio_venta = precio_venta;
    }
}
```

# PANTALLAS

## LOGIN

```
package FRAMES;

import CLASES.CONECCION2;

public class Login extends javax.swing.JInternalFrame {

    public String user="";
    public String passw="";

    public Login() {
        initComponents();
    }

private void jBAceptarActionPerformed(java.awt.event.ActionEvent evt) {
        String usuario = this.jCBUsuario.getSelectedItem().toString();
        String clave = new String(txt_clave.getPassword());
        new CONECCION2(usuario, clave);
        if(CONECCION2.conexion != null) {
            this.dispose();
        }
}

private void jBCerrarActionPerformed(java.awt.event.ActionEvent evt) {
       System.exit(0);
}

private void formInternalFrameClosed(javax.swing.event.InternalFrameEvent evt) {
        //GUARDAR DATOS EN VARIABLE PUBLICAS
        user = this.jCBUsuario.getSelectedItem().toString();
        passw = this.txt_clave.getText();
        //************************************************
    }
}
```

## PRINCIPAL

```
package FRAMES;
import javax.swing.JOptionPane;

public class MDI extends javax.swing.JFrame {
    Login login;
    Producto Prod;
    Proveedores Pro;
    PROV MPro;
    CAT MCat;


    SUBCAT MSubCat;
    PROD MProd;
    FACTURAR_VENTA fv;
    FACTURAR_COMPRA fc;
    Categoria Cat;
    Subcategoria Subcat;
    PREC M;
```

```
    REPORTEPROVEEDOR RP;
    REPORTEPRODUCTO RPR;
    REPORTEPRODCOMBINADO RPC;
    REPORTEVENTAS RV;
    REPORTECOMPRAS RC;
    REPORTESELECCIONPRODUCTO RSP;
    Arqueo ARQ;
    ARQ Arq;
    VENTA venta;
    COMPRA compra;

    public MDI() {

        initComponents();

    }

private void jMFormularioProductoActionPerformed(java.awt.event.ActionEvent evt) {
      if(Prod != null)
          Prod.dispose();
        Prod = new Producto();
        jDesktopPane1.add(Prod);
        Prod.show();
}

    private void jMFormularioProveedorActionPerformed(java.awt.event.ActionEvent evt) {
        if(Pro != null)
            Pro.dispose();
        Pro = new Proveedores();
        jDesktopPane1.add(Pro);
        Pro.show();
}

    private void jMFormularioCategoriaProductoActionPerformed(java.awt.event.ActionEvent evt) {
     if(Cat != null)
          Cat.dispose();
        Cat = new Categoria();
        jDesktopPane1.add(Cat);
        Cat.show();
}

 private void jMFormularioSubcategoriaProductoActionPerformed(java.awt.event.ActionEvent evt)
{
        if(Subcat != null)
            Subcat.dispose();
        Subcat = new Subcategoria();
        jDesktopPane1.add(Subcat);
        Subcat.show();
}

    private void jMFormularioVentaActionPerformed(java.awt.event.ActionEvent evt) {
       if(fv != null)
          fv.dispose();
        fv = new FACTURAR_VENTA();
        jDesktopPane1.add(fv);
        fv.show();
```

```
}

    private void jMFormularioCompraActionPerformed(java.awt.event.ActionEvent evt) {
        if(fc != null)
            fc.dispose();
        fc = new FACTURAR_COMPRA();
        jDesktopPane1.add(fc);
        fc.show();
}

    private void jMMostrarActionPerformed(java.awt.event.ActionEvent evt) {

}

    private void jMReporteProductoActionPerformed(java.awt.event.ActionEvent evt) {
 if(RPR != null)
            RPR.dispose();
        RPR = new REPORTEPRODUCTO();
        jDesktopPane1.add(RPR);
        RPR.show();
        RPR.run();
}

    private void jMReporteVentaActionPerformed(java.awt.event.ActionEvent evt) {
        if(RV != null)
            RV.dispose();
        RV = new REPORTEVENTAS();
        jDesktopPane1.add(RV);
        RV.show();
        RV.run();
}

    private void jMReporteCompraActionPerformed(java.awt.event.ActionEvent evt) {
        if(RC != null)
            RC.dispose();
        RC = new REPORTECOMPRAS();
        jDesktopPane1.add(RC);
        RC.show();
        RC.run();
}

    private void jMReporteProveedorActionPerformed(java.awt.event.ActionEvent evt) {
        if(RP != null)
            RP.dispose();
        RP = new REPORTEPROVEEDOR();
        jDesktopPane1.add(RP);
        RP.show();
        RP.run();
}

    private void jMMostrarProveedorActionPerformed(java.awt.event.ActionEvent evt) {
       if(MPro != null)
            MPro.dispose();
        MPro = new PROV();
        jDesktopPane1.add(MPro);
        MPro.show();
```

```
}

    private void jMInformacionDiseñadoresActionPerformed(java.awt.event.ActionEvent evt) {
          if(M != null)
           M.dispose();
        M = new PREC();
        jDesktopPane1.add(M);
        M.show();
}

    private void jMReporteCategoriaProductoActionPerformed(java.awt.event.ActionEvent evt) {
       if(RPC != null)
           RPC.dispose();
        RPC = new REPORTEPRODCOMBINADO();
        jDesktopPane1.add(RPC);
        RPC.show();
        RPC.run();
}

    private void jMReporteConsultaProductoActionPerformed(java.awt.event.ActionEvent evt) {
        if(RSP != null)
        RSP.dispose();
        String cod = JOptionPane.showInputDialog(null,"DIGITE EL CODIGO DEL PRODUCTO");
        RSP = new REPORTESELECCIONPRODUCTO(cod);
        jDesktopPane1.add(RSP);
        RSP.show();
        RSP.run();
}

    private void jDesktopPane1PropertyChange(java.beans.PropertyChangeEvent evt) {
        login = new Login();
        jDesktopPane1.add(login);
        login.show();
    }

    private void jMMostrarCategoriaActionPerformed(java.awt.event.ActionEvent evt) {
        if(MCat != null)
            MCat.dispose();
        MCat = new CAT();
        jDesktopPane1.add(MCat);
        MCat.show();
}

    private void jMMostrarSubcategoriaActionPerformed(java.awt.event.ActionEvent evt) {
         if(MSubCat != null)
           MSubCat.dispose();
        MSubCat = new SUBCAT();
        jDesktopPane1.add(MSubCat);
        MSubCat.show();
}

    private void jMMostrarProductoActionPerformed(java.awt.event.ActionEvent evt) {
       if(MProd != null)
           MProd.dispose();
        MProd = new PROD();
        jDesktopPane1.add(MProd);
```

```
        MProd.show();
}

    private void jMFormularioArqueoActionPerformed(java.awt.event.ActionEvent evt) {
         if(ARQ != null)
            ARQ.dispose();
        ARQ = new Arqueo();

        jDesktopPane1.add(ARQ);
        ARQ.show();
}

    private void jMMostrarVentaActionPerformed(java.awt.event.ActionEvent evt) {
        if(venta != null)
            venta.dispose();
        venta = new VENTA();
        jDesktopPane1.add(venta);
        venta.show();
}

    private void jMMostrarCompraActionPerformed(java.awt.event.ActionEvent evt) {
        if(compra != null)
            compra.dispose();
        compra = new COMPRA();
        jDesktopPane1.add(compra);
        compra.show();
}

    private void formPropertyChange(java.beans.PropertyChangeEvent evt) {
       this.DESACTIVARMENUS();
    }

    private void formFocusGained(java.awt.event.FocusEvent evt) {

    }

    private void jMMostrarArqueoActionPerformed(java.awt.event.ActionEvent evt) {
        if(Arq != null)
            Arq.dispose();
        Arq = new ARQ();
        jDesktopPane1.add(Arq);
        Arq.show();
    }

private void jBActivarMenuActionPerformed(java.awt.event.ActionEvent evt) {
   //ACTIVAR MENU DE LA PANTALLA PRINCIPAL
   String USER = login.user;
   String PASSW = login.passw;

    if(USER=="")
    {
        this.DESACTIVARMENUS();
    }

    if(USER=="Administrador")
    {
```

```
    this.ACTIVARMENUS1();
 }

 if(USER=="Facturador")
 {
    this.ACTIVARMENUS1();
 }

if(USER=="Bodega")
 {
    this.ACTIVARMENUS2();
 }

if(USER=="Propietario")
 {
    this.ACTIVARMENUS3();
 }

if(USER=="Vendedor")
 {
    this.ACTIVARMENUS4();
 }

if(USER!="")
 {
    this.jPanel1.setVisible(false);
 }
 }

 public void DESACTIVARMENUS(){
    this.jMFormularios.enable(false);
    this.jMReportes.enable(false);
    this.jMMostrar.enable(false);
    this.jMInformacion.enable(false);
 }

 public void ACTIVARMENUS1(){
    this.jMFormularios.enable(true);
    this.jMReportes.enable(true);
    this.jMMostrar.enable(true);
    this.jMInformacion.enable(true);
 }

 public void ACTIVARMENUS2(){
    this.jMFormularios.enable(true);
        this.jMFormularioSubcategoriaProducto.enable(true);
        this.jMFormularioCategoriaProducto.enable(true);
        this.jMFormularioProducto.enable(true);
        this.jMFormularioProveedor.enable(false);
        this.jMFormularioVenta.enable(false);
        this.jMFormularioCompra.enable(false);
        this.jMFormularioArqueo.enable(false);

    this.jMReportes.enable(false);

    this.jMMostrar.enable(true);
```

```
            this.jMMostrarCategoria.enable(true);


           this.jMMostrarSubcategoria.enable(true);
            this.jMMostrarProducto.enable(true);
            this.jMMostrarProveedor.enable(false);
            this.jMMostrarVenta.enable(false);
            this.jMMostrarCompra.enable(false);
            this.jMMostrarArqueo.enable(false);

        this.jMInformacion.enable(true);
    }

    public void ACTIVARMENUS3(){
        this.jMFormularios.enable(false);
        this.jMReportes.enable(true);
        this.jMMostrar.enable(true);
        this.jMInformacion.enable(true);
    }

    public void ACTIVARMENUS4(){
        this.jMFormularios.enable(false);

        this.jMReportes.enable(false);

        this.jMMostrar.enable(true);
            this.jMMostrarCategoria.enable(true);
            this.jMMostrarSubcategoria.enable(true);
            this.jMMostrarProducto.enable(true);
            this.jMMostrarProveedor.enable(false);
            this.jMMostrarVenta.enable(false);
            this.jMMostrarCompra.enable(false);
            this.jMMostrarArqueo.enable(false);

        this.jMInformacion.enable(true);
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MDI().setVisible(true);
            }
        });
    }
}
```

## VENTA
## PANTALLA1

```
package FRAMES;

import CLASES.CONECCION;
import CLASES.DETALLEVENTA;
import CLASES.PRODUCTO;
import CLASES.VENTA;
import CLASES.Validacion;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.text.DefaultFormatterFactory;
import javax.swing.text.MaskFormatter;

public class FACTURAR_VENTA extends javax.swing.JInternalFrame {
       ArrayList<VENTA> datos = new ArrayList<VENTA>();
    private int filas = -1;

    ArrayList<DETALLEVENTA> datos1 = new ArrayList<DETALLEVENTA>();
    private int filas1 = -1;

     CONECCION  mic = null;
     ResultSet result = null;
     PRODUCTO pro = null;

    public FACTURAR_VENTA() {
        initComponents();
    }

   private void jBAgregarFilaActionPerformed(java.awt.event.ActionEvent evt) {
     try
        {
        String cod = this.jTFCodigo.getText();
        String fecha = this.jFTFFecha.getText();
        String desc = this.jTFDescripcion.getText();
        String nombcli = this.jTFNomCliente.getText();
        double subt = Double.parseDouble(this.jTFSubtotal.getText());
        double iva = Double.parseDouble(this.jTFIva.getText());
        double total = Double.parseDouble(this.jTFTotal.getText());
        double pago = Double.parseDouble(this.jTFPago.getText());
        double vuelto = Double.parseDouble(this.jTFVuelto.getText());
        addRow();
        datos.add(new VENTA(cod,fecha,desc,nombcli,subt,iva,total,pago,vuelto));

        this.jTable1.setValueAt(datos.get(filas).getCod_factura(), filas, 0);
        this.jTable1.setValueAt(datos.get(filas).getFecha(), filas, 1);
        this.jTable1.setValueAt(datos.get(filas).getDescripcion(), filas, 2);
        this.jTable1.setValueAt(datos.get(filas).getCliente(), filas, 3);
        this.jTable1.setValueAt(datos.get(filas).getSubtotal(), filas, 4);
        this.jTable1.setValueAt(datos.get(filas).getIva(), filas, 5);
        this.jTable1.setValueAt(datos.get(filas).getTotal(), filas, 6);
        this.jTable1.setValueAt(datos.get(filas).getPago(), filas, 7);
        this.jTable1.setValueAt(datos.get(filas).getVuelto(), filas, 8);

        this.jTFCodigo.setText("");
        this.jFTFFecha.setText("");
        this.jTFDescripcion.setText("");
        this.jTFNomCliente.setText("");
        this.jTFSubtotal.setText("");
```

```
        this.jTFIva.setText("");
        this.jTFTotal.setText("");
        this.jTFVuelto.setText("");
        this.jTFPago.setText("");
        this.jTFCodigo.requestFocus();

        }

        catch(Exception ex)
        {
            JOptionPane.showMessageDialog(null,"Debe llenar todos los campos");
        }

}

    private void jBAgregarFilaDetalleActionPerformed(java.awt.event.ActionEvent evt) {
     try
        {
        String codf = this.jTextField1.getText();
        String codp = this.jTextField2.getText();
        int cant = Integer.parseInt(this.jTextField3.getText());

        double precio = Double.parseDouble(this.jTextField4.getText());
        double subtotal = Double.parseDouble(this.jTextField5.getText());
        double descuento = Double.parseDouble(this.jTextField6.getText());
        double tot = Double.parseDouble(this.jTextField7.getText());

        addRow1();
        datos1.add(new DETALLEVENTA(codf,codp,cant,precio,subtotal,descuento,tot));

        this.jTable2.setValueAt(datos1.get(filas1).getCod_factura(), filas1, 0);
        this.jTable2.setValueAt(datos1.get(filas1).getCod_producto(), filas1, 1);
        this.jTable2.setValueAt(datos1.get(filas1).getCantidad(), filas1, 2);
        this.jTable2.setValueAt(datos1.get(filas1).getPrecio(), filas1, 3);
        this.jTable2.setValueAt(datos1.get(filas1).getSubtotal(), filas1, 4);
        this.jTable2.setValueAt(datos1.get(filas1).getDescuento(), filas1, 5);
        this.jTable2.setValueAt(datos1.get(filas1).getTotal(), filas1, 6);

        this.jTextField1.setText("");
        this.jTextField2.setText("");
        this.jTextField3.setText("");
        this.jTextField4.setText("");
        this.jTextField5.setText("");
        this.jTextField6.setText("");
        this.jTextField7.setText("");

        this.jTextField1.requestFocus();

        }
        catch(Exception ex2)
        {
            JOptionPane.showMessageDialog(null,"Debe llenar todos los campos");
        }

}
```

```
    private void jBGuardarActionPerformed(java.awt.event.ActionEvent evt) {
        agregarDatos();
        agregarDatos1();
}

    private void jBCalcularTotalFacturaActionPerformed(java.awt.event.ActionEvent evt) {
        double num, cal, i, to, p;
        cal = Double.valueOf(this.jTFSubtotal.getText());
        num = Double.valueOf(this.jTextField7.getText());
        cal = cal+num;


 this.jTFSubtotal.setText(String.valueOf(cal));

        i = Double.valueOf(this.jTFSubtotal.getText())*0.15;
        to = cal+i;

        this.jTFIva.setText(String.valueOf(i));
        this.jTFTotal.setText(String.valueOf(to));
}

    private void jBCalcularTotalDetalleActionPerformed(java.awt.event.ActionEvent evt) {
        int c;
        double p, st, desc, t;
        c = Integer.valueOf(this.jTextField3.getText());
        p = Double.valueOf(this.jTextField4.getText());
        st = c*p;
        desc = st*0.10;
        t = st-desc;
        this.jTextField5.setText(String.valueOf(st));
        this.jTextField6.setText(String.valueOf(desc));
        this.jTextField7.setText(String.valueOf(t));
}

    private void jTextField3KeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTextField3.getText(),evt.getKeyChar());
        if(!val.validarEntero())
            evt.consume();
    }

    private void jTextField4KeyTyped(java.awt.event.KeyEvent evt) {
       Validacion val = new Validacion(this.jTextField4.getText(),evt.getKeyChar());
       if(!val.validarFloat())
           evt.consume();
    }

    private void jTextField5KeyTyped(java.awt.event.KeyEvent evt) {
       Validacion val = new Validacion(this.jTextField5.getText(),evt.getKeyChar());
       if(!val.validarFloat())
           evt.consume();
    }

    private void jTextField6KeyTyped(java.awt.event.KeyEvent evt) {
       Validacion val = new Validacion(this.jTextField6.getText(),evt.getKeyChar());
       if(!val.validarFloat())
           evt.consume();
```

```
    }

    private void jTextField7KeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTextField7.getText(),evt.getKeyChar());
        if(!val.validarFloat())
            evt.consume();
    }

    private void jTFSubtotalKeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTFSubtotal.getText(),evt.getKeyChar());
        if(!val.validarFloat())
            evt.consume();
    }

    private void jTFIvaKeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTFIva.getText(),evt.getKeyChar());
        if(!val.validarFloat())
            evt.consume();
    }

    private void jTFTotalKeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTFTotal.getText(),evt.getKeyChar());
        if(!val.validarFloat())
            evt.consume();
    }

    private void jTFPagoKeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTFPago.getText(),evt.getKeyChar());
        if(!val.validarFloat())
            evt.consume();
    }

    private void jTFVueltoKeyTyped(java.awt.event.KeyEvent evt) {
         Validacion val = new Validacion(this.jTFVuelto.getText(),evt.getKeyChar());
        if(!val.validarFloat())
            evt.consume();
    }


    private void jBEliminarFilaDetalleActionPerformed(java.awt.event.ActionEvent evt) {
        eliminarFila1(this.jTable2);
}

    private void jBAnularActionPerformed(java.awt.event.ActionEvent evt) {
        eliminarDatos();
        eliminarDatos1();
        this.jTFCodigo.setText("");
        this.jTFCodigo.requestFocus();
        JOptionPane.showMessageDialog(null,"LA FACTURA FUE ELIMINADA
SATISFACTORIAMENTE");
}

    private void jTextField4FocusGained(java.awt.event.FocusEvent evt) {
        showDatos();
    }
```

```
    private void jTextField3FocusLost(java.awt.event.FocusEvent evt) {
        compararCant();
    }

    private void jBCalcularVueltoActionPerformed(java.awt.event.ActionEvent evt) {
    double to, p, v;
    to = Double.valueOf(this.jTFTotal.getText());
    p = Double.valueOf(this.jTFPago.getText());
    v = p-to;

    this.jTFVuelto.setText(String.valueOf(v));
}

    private void jBSalirActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
}

    private void addRow()
{
  DefaultTableModel model = (DefaultTableModel) this.jTable1.getModel();
  Vector fila = new Vector();
  fila.add(null);
  model.addRow(fila);
  jTable1.setModel(model);
  filas = filas + 1;
}

  private void addRow1()
{
  DefaultTableModel model1 = (DefaultTableModel) this.jTable1.getModel();
  Vector fila = new Vector();
  fila.add(null);
  model1.addRow(fila);
  jTable1.setModel(model1);
  filas1 = filas1 + 1;
}

public void eliminarFila(JTable tabla)
    {
      DefaultTableModel mod = (DefaultTableModel) tabla.getModel();
      int indice = tabla.getSelectedRow();
      if(indice!=-1)
      {
        mod.removeRow(indice);
        datos.remove(indice);
        filas = filas - 1;
      }
    }
    public void eliminarFila(JTable tabla, int row)
    {
      DefaultTableModel mod = (DefaultTableModel) tabla.getModel();
      int indice = row;
      if(indice!=-1)
      {
        mod.removeRow(indice);
        datos.remove(indice);
```

```
        filas = filas - 1;
      }
    }

    public void eliminarFila1(JTable tabla)
    {
      DefaultTableModel mod1 = (DefaultTableModel) tabla.getModel();
      int indice1 = tabla.getSelectedRow();
      if(indice1!=-1)
      {
        mod1.removeRow(indice1);
        datos1.remove(indice1);
        filas1 = filas1 - 1;
      }
    }

    public void eliminarFila1(JTable tabla, int row1)
    {
      DefaultTableModel mod1 = (DefaultTableModel) tabla.getModel();
      int indice1 = row1;
      if(indice1!=-1)
      {
        mod1.removeRow(indice1);
        datos1.remove(indice1);
        filas1 = filas1 - 1;
      }
    }

     private void agregarDatos()
    {
      try {
          CONECCION con = new CONECCION();
          String sql = "INSERT INTO TIENDA..FACTURAR_VENTA(Cod_Factura_Venta,
Fecha_Venta, Descripcion, Nombre_Cliente," +
              "Subtotal, IVA, Total, Pago, Vuelto) VALUES(?,?,?,?,?,?,?,?,?)";
          con.pConsulta = con.conexion.prepareStatement(sql);

          int i=0;
          while(datos.size() != 0)
          {
            con.pConsulta.setString(1, datos.get(i).getCod_factura());
            con.pConsulta.setString(2, datos.get(i).getFecha());
            con.pConsulta.setString(3, datos.get(i).getDescripcion());
            con.pConsulta.setString(4, datos.get(i).getCliente());
            con.pConsulta.setDouble(5, datos.get(i).getSubtotal());
            con.pConsulta.setDouble(6, datos.get(i).getIva());
            con.pConsulta.setDouble(7, datos.get(i).getTotal());
            con.pConsulta.setDouble(8, datos.get(i).getPago());
            con.pConsulta.setDouble(9, datos.get(i).getVuelto());
            con.pConsulta.executeUpdate();
            eliminarFila(jTable1,i);

          }
          con.conexion.close();
       } catch (SQLException ex) {
```

```
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden ingresar datos");
        }
        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden ingresar datos");
                }
    }

    private void agregarDatos1()
    {
        try {
            CONECCION con = new CONECCION();
            String sql1 = "INSERT INTO
TIENDA..DETALLE_FACTURA_VENTA(Cod_Factura_Venta, Cod_Producto, Cantidad, Precio,"
+
                "Subtotal, Descuento, Total) VALUES(?,?,?,?,?,?,?)";
            con.pConsulta = con.conexion.prepareStatement(sql1);

            int i=0;
            while(datos1.size() != 0)
            {
              con.pConsulta.setString(1, datos1.get(i).getCod_factura());
              con.pConsulta.setString(2, datos1.get(i).getCod_producto());
              con.pConsulta.setInt(3, datos1.get(i).getCantidad());
              con.pConsulta.setDouble(4, datos1.get(i).getPrecio());
              con.pConsulta.setDouble(5, datos1.get(i).getSubtotal());
              con.pConsulta.setDouble(6, datos1.get(i).getDescuento());
              con.pConsulta.setDouble(7, datos1.get(i).getTotal());
              con.pConsulta.executeUpdate();
              eliminarFila1(jTable2,i);
            }

            con.conexion.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden ingresar datos");
        }
        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden ingresar datos");
                }
    }

    private void eliminarDatos()
    {
        try {
            CONECCION con = new CONECCION();
            String cod = this.jTFCodigo.getText();
            String sql = "DELETE FROM TIENDA..DETALLE_FACTURA_VENTA WHERE
Cod_Factura_Venta LIKE ?";

            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);
```

```
            del.setString(1, cod);
            del.executeUpdate();
            con.conexion.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se puede eliminar datos");
        }
        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
puede eliminar datos");
                }
    }

private void eliminarDatos1()
    {
        try {
            CONECCION con = new CONECCION();
            String cod = this.jTFCodigo.getText();
            String sql = "DELETE FROM TIENDA..FACTURAR_VENTA WHERE Cod_Factura_Venta
LIKE ?";

            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

            del.setString(1, cod);
            del.executeUpdate();
            con.conexion.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se puede eliminar datos");
        }
        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
puede eliminar datos");
                }
    }

private void showDatos()
    {
        try {
            CONECCION con = new CONECCION();
            String cod = this.jTextField2.getText();
            double precio;
            String sql = "SELECT Precio_Venta FROM TIENDA..PRODUCTO WHERE
Cod_Producto LIKE ?";

            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

            del.setString(1, cod);
            del.executeQuery();
            result = del.getResultSet();
            while (result.next()) {
                precio = result.getDouble(1);
                this.jTextField4.setText(String.valueOf(precio));
            }
```

```
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
        }
        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
                }
    }

private void compararCant()
    {
        try {
            CONECCION con = new CONECCION();
            String cod = this.jTextField2.getText();
            int cant, comptf;

            String sql = "SELECT Existencia FROM TIENDA..PRODUCTO WHERE Cod_Producto
LIKE ?";
            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);
            del.setString(1, cod);
            del.executeQuery();
            result = del.getResultSet();
            while (result.next()) {
                cant = result.getInt(1);
                comptf = Integer.valueOf(this.jTextField3.getText());
                if(comptf>cant)
                {
                JOptionPane.showMessageDialog(null, "LA CANTIDAD PEDIDA EXCEDE LA
CANTIDAD EN EXISTENCIA");
                }
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
        }
        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
                }
    }
}
```

## PANTALLA2

```
package FRAMES;

import CLASES.CONECCION;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

```
public class VENTA extends javax.swing.JInternalFrame {

    ResultSet result = null;

    public VENTA() {
        initComponents();
    }

    private void jBBuscarFacturaActionPerformed(java.awt.event.ActionEvent evt) {
        consulta();
}

    private void formPropertyChange(java.beans.PropertyChangeEvent evt) {
        showDatos();
    }

    private void jBMostrarDetalleActionPerformed(java.awt.event.ActionEvent evt) {
        consulta_detalle();
}

    private void jBSalirActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
}

private void addRow(){
        DefaultTableModel model = (DefaultTableModel)jTable1.getModel();
        Vector fila = new Vector();
        fila.add(null);
        model.addRow(fila);
        jTable1.setModel(model);
    }


 private void addRow1(){
        DefaultTableModel model = (DefaultTableModel)jTable2.getModel();
        Vector fila = new Vector();
        fila.add(null);
        model.addRow(fila);
        jTable2.setModel(model);
    }

private void showDatos()
    {
        try {
            String CodFactura="";
            String Fecha="";
            String Descripcion="";
            String NombreCliente="";
            Double Subtotal=0.0;


            Double IVA=0.0;
            Double Total=0.0;
            Double Pago=0.0;
            Double Vuelto=0.0;
```

```
        CONECCION con = new CONECCION();
        String sql = "SELECT * FROM TIENDA..FACTURAR_VENTA";
        java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

        del.executeQuery();
        result = del.getResultSet();

int i = 0;

        while (result.next()) {
           CodFactura = result.getString(1);
           Fecha = result.getString(2);
           Descripcion = result.getString(3);
           NombreCliente = result.getString(4);
           Subtotal = result.getDouble(5);
           IVA = result.getDouble(6);
           Total = result.getDouble(7);
           Pago = result.getDouble(8);
           Vuelto = result.getDouble(9);
  addRow();
   jTable1.setValueAt(result.getString(1),i, 0);
   jTable1.setValueAt(result.getString(2),i,1);
   jTable1.setValueAt(result.getString(3),i,2);
   jTable1.setValueAt(result.getString(4),i,3);
   jTable1.setValueAt(result.getDouble(5),i,4);
   jTable1.setValueAt(result.getDouble(6),i,5);
   jTable1.setValueAt(result.getDouble(7),i,6);
   jTable1.setValueAt(result.getDouble(8),i,7);
   jTable1.setValueAt(result.getDouble(9),i,8);

   i++;

      }
        con.conexion.close();
      }
        catch (SQLException ex) {
        JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
   }

private void consulta()
   {
      try {
        String CodFactura="";
        String Fecha="";
        String Descripcion="";
        String NombreCliente="";

        Double Subtotal=0.0;
        Double IVA=0.0;
```

```
        Double Total=0.0;
        Double Pago=0.0;
        Double Vuelto=0.0;

        CONECCION con = new CONECCION();
        String cod = JOptionPane.showInputDialog(null,"DIGITE EL CODIGO DE LA
FACTURA");
        String sql = "SELECT * FROM TIENDA..FACTURAR_VENTA WHERE
Cod_Factura_Venta LIKE ?";
        java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

        del.setString(1, cod);
        del.executeQuery();
        result = del.getResultSet();

        while (result.next()) {
          CodFactura = result.getString(1);
           Fecha = result.getString(2);
           Descripcion = result.getString(3);
           NombreCliente = result.getString(4);
           Subtotal = result.getDouble(5);
           IVA = result.getDouble(6);
           Total = result.getDouble(7);
           Pago = result.getDouble(8);
           Vuelto = result.getDouble(9);

        this.jTFCodFactura.setText(String.valueOf(CodFactura));
        this.jTFFechaVenta.setText(String.valueOf(Fecha));
        this.jTFDescripcion.setText(String.valueOf(Descripcion));
        this.jTFNombreCliente.setText(String.valueOf(NombreCliente));
        this.jTFSubTotal.setText(String.valueOf(Subtotal));
        this.jTFIva.setText(String.valueOf(IVA));
        this.jTFTotal.setText(String.valueOf(Total));
        this.jTFPago.setText(String.valueOf(Pago));
        this.jTFVuelto.setText(String.valueOf(Vuelto));

      }
        con.conexion.close();
      }

        catch (SQLException ex) {
        JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
   }

private void consulta_detalle()
   {
      try {
        String ID="";
        String CodFactura="";
```

```
        String CodProducto="";
        Integer Cantidad=0;
        Double Precio=0.0;


        Double Subtotal=0.0;
        Double Descuento=0.0;
        Double Total=0.0;

        CONECCION con = new CONECCION();
        String cod = this.jTFCodFactura.getText();
        String sql = "SELECT * FROM TIENDA..DETALLE_FACTURA_VENTA WHERE
Cod_Factura_Venta LIKE ?";
        java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

        del.setString(1, cod);
        del.executeQuery();
        result = del.getResultSet();

      int i = 0;
        while (result.next()) {
          ID = result.getString(1);
          CodFactura = result.getString(2);
          CodProducto = result.getString(3);
          Cantidad = result.getInt(4);
          Precio = result.getDouble(5);
          Subtotal = result.getDouble(6);
          Descuento = result.getDouble(7);
          Total = result.getDouble(8);

        addRow1();
           jTable2.setValueAt(result.getString(1),i, 0);
           jTable2.setValueAt(result.getString(2),i, 1);
           jTable2.setValueAt(result.getString(3),i,2);
           jTable2.setValueAt(result.getInt(4),i,3);
           jTable2.setValueAt(result.getDouble(5),i,4);
           jTable2.setValueAt(result.getDouble(6),i,5);
           jTable2.setValueAt(result.getDouble(7),i,6);
           jTable2.setValueAt(result.getDouble(8),i,7);

          i++;
      }

        con.conexion.close();
      }
        catch (SQLException ex) {
        JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
   }
}
```

## COMPRA
## PANTALLA1

```
package FRAMES;

import CLASES.COMPRA;
import CLASES.CONECCION;
import CLASES.DETALLECOMPRA;
import CLASES.PRODUCTO;
import CLASES.Validacion;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.text.DefaultFormatterFactory;
import javax.swing.text.MaskFormatter;

public class FACTURAR_COMPRA extends javax.swing.JInternalFrame {
       ArrayList<COMPRA> datos = new ArrayList<COMPRA>();
    private int filas = -1;

    ArrayList<DETALLECOMPRA> datos1 = new ArrayList<DETALLECOMPRA>();
    private int filas1 = -1;

 CONECCION  mic = null;
     ResultSet result = null;
     PRODUCTO pro = null;

    public FACTURAR_COMPRA() {
        initComponents();
    }

    private void jBAgregarFilaActionPerformed(java.awt.event.ActionEvent evt) {
     try
        {
        String codf = this.jTFCodigo.getText();
        String codp = this.jTFCodProveedor.getText();
        String fecha = this.jFTFFecha.getText();
        double total = Double.parseDouble(this.jTFTotal.getText());

        addRow();
        datos.add(new COMPRA(codf,codp,fecha,total));

        this.jTable1.setValueAt(datos.get(filas).getCod_factura(), filas, 0);
        this.jTable1.setValueAt(datos.get(filas).getCod_proveedor(), filas, 1);
        this.jTable1.setValueAt(datos.get(filas).getFecha(), filas, 2);
        this.jTable1.setValueAt(datos.get(filas).getTotal(), filas, 3);

        this.jTFCodigo.setText("");
        this.jTFCodProveedor.setText("");
        this.jFTFFecha.setText("");
```

```
        this.jTFTotal.setText("");

        this.jTFCodigo.requestFocus();

        }
        catch(Exception ex)
        {
            JOptionPane.showMessageDialog(null,"Debe llenar todos los campos");
        }

}

    private void jBAgregarFilaDetalleActionPerformed(java.awt.event.ActionEvent evt) {
      try
        {
        String codf = this.jTextField1.getText();
        String codp = this.jTextField2.getText();
        String prod = this.jTextField3.getText();
        int cant = Integer.parseInt(this.jTextField4.getText());

        double precio = Double.parseDouble(this.jTextField5.getText());
        double subtotal = Double.parseDouble(this.jTextField6.getText());

        addRow1();
        datos1.add(new DETALLECOMPRA(codf,codp,prod,cant,precio,subtotal));

        this.jTable2.setValueAt(datos1.get(filas1).getCod_factura(), filas1, 0);
        this.jTable2.setValueAt(datos1.get(filas1).getCod_producto(), filas1, 1);
        this.jTable2.setValueAt(datos1.get(filas1).getProducto(), filas1, 2);
        this.jTable2.setValueAt(datos1.get(filas1).getCantidad(), filas1, 3);
        this.jTable2.setValueAt(datos1.get(filas1).getPrecio(), filas1, 4);
        this.jTable2.setValueAt(datos1.get(filas1).getTotal(), filas1, 5);

        this.jTextField1.setText("");
        this.jTextField2.setText("");
        this.jTextField3.setText("");
        this.jTextField4.setText("");
        this.jTextField5.setText("");
        this.jTextField6.setText("");

        this.jTextField1.requestFocus();

        }
        catch(Exception ex2)
        {
            JOptionPane.showMessageDialog(null,"Debe llenar todos los campos");
        }

}

    private void jBGuardarActionPerformed(java.awt.event.ActionEvent evt) {
        agregarDatos();
        agregarDatos1();
}

    private void jBCalcularTotalFacturaActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        double num, cal;
        cal = Double.valueOf(this.jTFTotal.getText());
        num = Double.valueOf(this.jTextField6.getText());
        cal = cal+num;
        this.jTFTotal.setText(String.valueOf(cal));
}

    private void jBCalcularSubtotalActionPerformed(java.awt.event.ActionEvent evt) {
        int c;
        double p, t;
        c = Integer.valueOf(this.jTextField4.getText());
        p = Double.valueOf(this.jTextField5.getText());
        t = c*p;
        this.jTextField6.setText(String.valueOf(t));
}

    private void jTFTotalKeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTFTotal.getText(),evt.getKeyChar());
       if(!val.validarFloat())
          evt.consume();
    }

    private void jTextField5KeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTextField5.getText(),evt.getKeyChar());
       if(!val.validarFloat())
          evt.consume();
    }

    private void jTextField6KeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTextField6.getText(),evt.getKeyChar());
       if(!val.validarFloat())
          evt.consume();
    }

    private void jTextField4KeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTextField4.getText(),evt.getKeyChar());
        if(!val.validarEntero())
           evt.consume();
    }

    private void jTextField5FocusGained(java.awt.event.FocusEvent evt) {
        showDatos();
    }

    private void jBAnularActionPerformed(java.awt.event.ActionEvent evt) {
        eliminarDatos();
        eliminarDatos1();
        this.jTFCodigo.setText("");
        this.jTFCodigo.requestFocus();
        JOptionPane.showMessageDialog(null,"LA FACTURA FUE ELIMINADA
SATISFACTORIAMENTE");
}

    private void jBEliminarFilaDetalleActionPerformed(java.awt.event.ActionEvent evt) {
         eliminarFila1(this.jTable2);
}
```

```
    private void jBSalirActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
}

    private void addRow()
{
  DefaultTableModel model = (DefaultTableModel) this.jTable1.getModel();
  Vector fila = new Vector();
  fila.add(null);
  model.addRow(fila);
  jTable1.setModel(model);
  filas = filas + 1;
}

  private void addRow1()
{
  DefaultTableModel model1 = (DefaultTableModel) this.jTable1.getModel();
  Vector fila = new Vector();
  fila.add(null);
  model1.addRow(fila);
  jTable1.setModel(model1);
  filas1 = filas1 + 1;
}

public void eliminarFila(JTable tabla)
    {
      DefaultTableModel mod = (DefaultTableModel) tabla.getModel();
      int indice = tabla.getSelectedRow();
      if(indice!=-1)
      {
        mod.removeRow(indice);
        datos.remove(indice);
        filas = filas - 1;
      }
    }

    public void eliminarFila(JTable tabla, int row)
    {
      DefaultTableModel mod = (DefaultTableModel) tabla.getModel();
      int indice = row;
      if(indice!=-1)
      {
        mod.removeRow(indice);
        datos.remove(indice);
        filas = filas - 1;
      }
    }

    public void eliminarFila1(JTable tabla)
    {
      DefaultTableModel mod1 = (DefaultTableModel) tabla.getModel();
      int indice1 = tabla.getSelectedRow();
      if(indice1!=-1)
      {
```

```
        mod1.removeRow(indice1);
        datos1.remove(indice1);
        filas1 = filas1 - 1;
      }
   }
   public void eliminarFila1(JTable tabla, int row1)
   {
      DefaultTableModel mod1 = (DefaultTableModel) tabla.getModel();
      int indice1 = row1;
      if(indice1!=-1)
      {
        mod1.removeRow(indice1);
        datos1.remove(indice1);
        filas1 = filas1 - 1;
      }
   }

private void agregarDatos()
   {
       try {
          CONECCION con = new CONECCION();
          String sql = "INSERT INTO TIENDA..FACTURAR_COMPRA(Cod_Factura_Compra,
Cod_Proveedor, Fecha_Compra, " +
              "Total) VALUES(?,?,?,?)";
          con.pConsulta = con.conexion.prepareStatement(sql);

          int i=0;
          while(datos.size() != 0)
          {
            con.pConsulta.setString(1, datos.get(i).getCod_factura());
            con.pConsulta.setString(2, datos.get(i).getCod_proveedor());
            con.pConsulta.setString(3, datos.get(i).getFecha());
            con.pConsulta.setDouble(4, datos.get(i).getTotal());

            con.pConsulta.executeUpdate();
            eliminarFila(jTable1,i);
          }
          con.conexion.close();
       } catch (SQLException ex) {

            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden ingresar datos");
       }
       catch(NullPointerException ex)
              {
               JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden ingresar datos");
              }
   }

   private void agregarDatos1()
   {
       try {
          CONECCION con = new CONECCION();
```

```
            String sql1 = "INSERT INTO
TIENDA..DETALLE_FACTURA_COMPRA(Cod_Factura_Compra, Cod_Producto,
Producto,Cantidad, Precio," +
                "Subtotal) VALUES(?,?,?,?,?,?)";
            con.pConsulta = con.conexion.prepareStatement(sql1);

            int i=0;
            while(datos1.size() != 0)
            {
              con.pConsulta.setString(1, datos1.get(i).getCod_factura());
              con.pConsulta.setString(2, datos1.get(i).getCod_producto());
              con.pConsulta.setString(3, datos1.get(i).getProducto());
              con.pConsulta.setInt(4, datos1.get(i).getCantidad());
              con.pConsulta.setDouble(5, datos1.get(i).getPrecio());
              con.pConsulta.setDouble(6, datos1.get(i).getTotal());

              con.pConsulta.executeUpdate();
              eliminarFila1(jTable2,i);

            }
            con.conexion.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden ingresar datos");
        }
        catch(NullPointerException ex)
              {
               JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden ingresar datos");
              }
    }

    private void eliminarDatos()
    {
        try {
            CONECCION con = new CONECCION();
            String cod = this.jTFCodigo.getText();
            String sql = "DELETE FROM TIENDA..DETALLE_FACTURA_COMPRA WHERE
Cod_Factura_Compra LIKE ?";

            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

            del.setString(1, cod);
            del.executeUpdate();
            con.conexion.close();

        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se puede eliminar datos");
        }
        catch(NullPointerException ex)
              {
               JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
puede eliminar datos");
              }
    }
```

```
private void eliminarDatos1()
   {
      try {
         CONECCION con = new CONECCION();
         String cod = this.jTFCodigo.getText();
         String sql = "DELETE FROM TIENDA..FACTURAR_COMPRA WHERE
Cod_Factura_Compra LIKE ?";
         java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

         del.setString(1, cod);
         del.executeUpdate();
         con.conexion.close();
      } catch (SQLException ex) {
         JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se puede eliminar datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
puede eliminar datos");
            }
   }

private void showDatos()
   {
      try {
         CONECCION con = new CONECCION();
         String cod = this.jTextField2.getText();
         double precio;
         String sql = "SELECT Precio_Compra FROM TIENDA..PRODUCTO WHERE
Cod_Producto LIKE ?";

         java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

         del.setString(1, cod);
         del.executeQuery();
         result = del.getResultSet();
         while (result.next()) {
            precio = result.getDouble(1);
            this.jTextField5.setText(String.valueOf(precio));

         }
      } catch (SQLException ex) {
         JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
   }
}
```

## PANTALLA2

```
package FRAMES;

import CLASES.CONECCION;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Vector;
import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class COMPRA extends javax.swing.JInternalFrame {

     ResultSet result = null;


    public COMPRA() {
        initComponents();
    }

    private void jBBuscarFacturaActionPerformed(java.awt.event.ActionEvent evt) {
        consulta();
}

    private void jBMostrarDetalleActionPerformed(java.awt.event.ActionEvent evt) {
        consulta_detalle();
}

    private void formPropertyChange(java.beans.PropertyChangeEvent evt) {
        showDatos();
    }

    private void jBSalirActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
}

    private void addRow(){
        DefaultTableModel model = (DefaultTableModel)jTable1.getModel();
        Vector fila = new Vector();
        fila.add(null);
        model.addRow(fila);
        jTable1.setModel(model);
    }

 private void addRow1(){
        DefaultTableModel model = (DefaultTableModel)jTable2.getModel();
        Vector fila = new Vector();
        fila.add(null);
        model.addRow(fila);
        jTable2.setModel(model);
    }

private void showDatos()
    {
        try {
            String CodFactura="";
```

```
            String CodProducto="";
            String FechaCompra="";
            Double Total=0.0;

            CONECCION con = new CONECCION();
            String sql = "SELECT * FROM TIENDA..FACTURAR_COMPRA";
            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

            del.executeQuery();
            result = del.getResultSet();

int i = 0;

            while (result.next()) {
               CodFactura = result.getString(1);
               CodProducto = result.getString(2);
               FechaCompra = result.getString(3);
               Total = result.getDouble(4);


  addRow();
   jTable1.setValueAt(result.getString(1),i, 0);
   jTable1.setValueAt(result.getString(2),i,1);
   jTable1.setValueAt(result.getString(3),i,2);
   jTable1.setValueAt(result.getDouble(4),i,3);

   i++;

      }
            con.conexion.close();
      }
            catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
   }

private void consulta()
   {
      try {
            String CodFactura="";
            String CodProducto="";
            String FechaCompra="";
            Double Total=0.0;
            CONECCION con = new CONECCION();
            String cod = JOptionPane.showInputDialog(null,"DIGITE EL CODIGO DE LA
FACTURA");
            String sql = "SELECT * FROM TIENDA..FACTURAR_COMPRA WHERE
Cod_Factura_Compra LIKE ?";
            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);
```

```
            del.setString(1, cod);
            del.executeQuery();
            result = del.getResultSet();

            while (result.next()) {
                CodFactura = result.getString(1);
                CodProducto = result.getString(2);
                FechaCompra = result.getString(3);
                Total = result.getDouble(4);

            this.jTFCodFactura.setText(String.valueOf(CodFactura));
            this.jTFCodigoProducto.setText(String.valueOf(CodProducto));
            this.jTFFechaCompra.setText(String.valueOf(FechaCompra));
            this.jTFTotal.setText(String.valueOf(Total));

        }
            con.conexion.close();
        }

            catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
        }

        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
                }
    }

private void consulta_detalle()
    {
        try {
            String ID="";
            String CodFactura="";
            String CodProducto="";
            String Producto="";
            Integer Cantidad=0;
            Double Precio=0.0;
            Double Subtotal=0.0;

            CONECCION con = new CONECCION();
            String cod = this.jTFCodFactura.getText();
            String sql = "SELECT * FROM TIENDA..DETALLE_FACTURA_COMPRA WHERE
Cod_Factura_Compra LIKE ?";
            java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

            del.setString(1, cod);
            del.executeQuery();
            result = del.getResultSet();

        int i = 0;
            while (result.next()) {
              ID = result.getString(1);
              CodFactura = result.getString(2);
```

```
            CodProducto = result.getString(3);
            Producto = result.getString(4);
            Cantidad = result.getInt(5);
            Precio = result.getDouble(6);
            Subtotal = result.getDouble(7);

          addRow1();
             jTable2.setValueAt(result.getString(1),i, 0);
             jTable2.setValueAt(result.getString(2),i, 1);
             jTable2.setValueAt(result.getString(3),i,2);
             jTable2.setValueAt(result.getString(4),i,3);
             jTable2.setValueAt(result.getInt(5),i,4);
             jTable2.setValueAt(result.getDouble(6),i,5);
             jTable2.setValueAt(result.getDouble(7),i,6);

            i++;
      }
          con.conexion.close();
      }
          catch (SQLException ex) {
          JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }

      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
   }

}
```

## PRODUCTO
## PANTALLA1

```
package FRAMES;

import CLASES.CONECCION;
import CLASES.PRODUCTO;
import CLASES.Validacion;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class Producto extends javax.swing.JInternalFrame {
   ArrayList<PRODUCTO> datos = new ArrayList<PRODUCTO>();
   private int filas = -1;
   private int filas1 = -1;

   ResultSet result = null;
```

```
    public Producto() {
        initComponents();
    }

    private void jBAgregarFilaActionPerformed(java.awt.event.ActionEvent evt) {
        this.jTFCodigoProducto.getText();
      try
        {
        String codprod = this.jTFCodigoProducto.getText();
        String codcat = this.jTFCategoria.getText();
        String codsubc = this.jTFSubcategoria.getText();
        double precio_compra = Double.valueOf(this.jTFPrecioCompra.getText());
        double precio_venta = Double.valueOf(this.jTFPrecioVenta.getText());
        int existencia = Integer.valueOf(this.jTFExistencia.getText());

        addRow();
        datos.add(new
PRODUCTO(codprod,codcat,codsubc,precio_compra,precio_venta,existencia));

        this.jTable1.setValueAt(datos.get(filas).getCod_producto(), filas, 0);
        this.jTable1.setValueAt(datos.get(filas).getCod_categoria(), filas, 1);
        this.jTable1.setValueAt(datos.get(filas).getCod_subcategoria(), filas, 2);
        this.jTable1.setValueAt(datos.get(filas).getPrecio_compra(), filas, 3);
        this.jTable1.setValueAt(datos.get(filas).getPrecio_venta(), filas,4);
        this.jTable1.setValueAt(datos.get(filas).getExistencia(), filas, 5);
        this.jTFCodigoProducto.setText("");
        this.jTFCategoria.setText("");
        this.jTFSubcategoria.setText("");
        this.jTFPrecioCompra.setText("");
        this.jTFPrecioVenta.setText("");
        this.jTFExistencia.setText("");
        this.jTFCodigoProducto.requestFocus();

        }
        catch(Exception ex)
        {
            JOptionPane.showMessageDialog(null,"Debe llenar todos los campos");
        }
}

    private void jBGuardarActionPerformed(java.awt.event.ActionEvent evt) {
        agregarDatos();
    }

    private void jBEliminarActionPerformed(java.awt.event.ActionEvent evt) {
        eliminarDatos();
        this.jTFCodigoProducto.setText("");
        this.jTFCodigoProducto.requestFocus();
        JOptionPane.showMessageDialog(null,"EL PRODUCTO FUE ELIMINADO
SATISFACTORIAMENTE");
    }

    private void jBModificarActionPerformed(java.awt.event.ActionEvent evt) {
        actualizarDatos();
        this.jTFCodigoProducto.setText("");
        this.jTFCategoria.setText("");
```

```
        this.jTFSubcategoria.setText("");
        this.jTFPrecioCompra.setText("");
        this.jTFPrecioVenta.setText("");
        this.jTFExistencia.setText("");
        this.jTFCodigoProducto.requestFocus();
        JOptionPane.showMessageDialog(null,"EL PRODUCTO FUE MODIFICADO
SATISFACTORIAMENTE");
    }

    private void jTFExistenciaKeyTyped(java.awt.event.KeyEvent evt) {
        Validacion val = new Validacion(this.jTFExistencia.getText(),evt.getKeyChar());
        if(!val.validarEntero())
            evt.consume();
    }

    private void jTFPrecioCompraKeyTyped(java.awt.event.KeyEvent evt) {
       Validacion val = new Validacion(this.jTFPrecioCompra.getText(),evt.getKeyChar());
       if(!val.validarFloat())
           evt.consume();
    }

    private void jTFPrecioVentaKeyTyped(java.awt.event.KeyEvent evt) {
       Validacion val = new Validacion(this.jTFPrecioVenta.getText(),evt.getKeyChar());
       if(!val.validarFloat())
           evt.consume();
    }

    private void jBModificarFilaActionPerformed(java.awt.event.ActionEvent evt) {
       int indice = jTable1.getSelectedRow();
        if(indice!=-1)
        {
        this.jTFCodigoProducto.setText(String.valueOf(datos.get(indice).getCod_producto()));
        this.jTFCategoria.setText(String.valueOf(datos.get(indice).getCod_categoria()));
        this.jTFSubcategoria.setText(String.valueOf(datos.get(indice).getCod_subcategoria()));
        this.jTFPrecioCompra.setText(String.valueOf(datos.get(indice).getPrecio_compra()));
        this.jTFPrecioVenta.setText(String.valueOf(datos.get(indice).getPrecio_venta()));
        this.jTFExistencia.setText(String.valueOf(datos.get(indice).getExistencia()));
        eliminarFila(this.jTable1);
        this.jTFCodigoProducto.requestFocus();
        }
}

    private void jBEliminarFilaActionPerformed(java.awt.event.ActionEvent evt) {
        eliminarFila(this.jTable1);
}

    private void jBLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
        this.jTFCodigoProducto.setText("");
        this.jTFCategoria.setText("");
        this.jTFSubcategoria.setText("");
        this.jTFPrecioCompra.setText("");
        this.jTFPrecioVenta.setText("");
        this.jTFExistencia.setText("");
}

    private void jBActualizarActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        showDatos();
}

    private void formPropertyChange(java.beans.PropertyChangeEvent evt) {
        showDatos();
    }

    private void jBSalirActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
    }

private void addRow()
{
  DefaultTableModel model = (DefaultTableModel) this.jTable1.getModel();
  Vector fila = new Vector();
  fila.add(null);
  model.addRow(fila);
  jTable1.setModel(model);
  filas = filas + 1;
}


private void addRow1()
{
  DefaultTableModel model = (DefaultTableModel) this.jTable2.getModel();
  Vector fila = new Vector();
  fila.add(null);
  model.addRow(fila);
  jTable2.setModel(model);
  filas1 = filas1 + 1;
}

public void eliminarFila(JTable tabla)
    {
       DefaultTableModel mod = (DefaultTableModel) tabla.getModel();
       int indice = tabla.getSelectedRow();
       if(indice!=-1)
       {
          mod.removeRow(indice);
          datos.remove(indice);
          filas = filas - 1;
       }
    }
 public void eliminarFila(JTable tabla, int row)
    {
       DefaultTableModel mod = (DefaultTableModel) tabla.getModel();
       int indice = row;
       if(indice!=-1)
       {
          mod.removeRow(indice);
          datos.remove(indice);
          filas = filas - 1;
       }
    }

private void agregarDatos()
```

```
   {
      try {
         CONECCION con = new CONECCION();
         String sql = "INSERT INTO TIENDA..PRODUCTO(Cod_Producto, Cod_Categoria,
Cod_Subcategoria, Precio_Compra, Precio_Venta, Existencia) VALUES(?,?,?,?,?,?)";
         con.pConsulta = con.conexion.prepareStatement(sql);

         int i=0;
         while(datos.size() != 0)
         {
           con.pConsulta.setString(1, datos.get(i).getCod_producto());
           con.pConsulta.setString(2, datos.get(i).getCod_categoria());
           con.pConsulta.setString(3, datos.get(i).getCod_subcategoria());
           con.pConsulta.setDouble(4, datos.get(i).getPrecio_compra());
           con.pConsulta.setDouble(5, datos.get(i).getPrecio_venta());
           con.pConsulta.setInt(6, datos.get(i).getExistencia());

           con.pConsulta.executeUpdate();
           eliminarFila(jTable1,i);

         }
         con.conexion.close();
      } catch (SQLException ex) {
         JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden ingresar datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden ingresar datos");
            }
   }

private void eliminarDatos()
   {
      try {
         CONECCION con = new CONECCION();
         String cod = this.jTFCodigoProducto.getText();
         String sql = "DELETE FROM TIENDA..PRODUCTO WHERE Cod_Producto LIKE ?";

         java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

         del.setString(1, cod);
         del.executeUpdate();
         con.conexion.close();
      } catch (SQLException ex) {
         JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se puede eliminar datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
puede eliminar datos");
            }
   }
```

```
private void actualizarDatos()
   {
      try {
         CONECCION con = new CONECCION();
         String codp = this.jTFCodigoProducto.getText();
         String codc = this.jTFCategoria.getText();
         String cods = this.jTFSubcategoria.getText();
         double pc = Double.valueOf(this.jTFPrecioCompra.getText());
         double pv = Double.valueOf(this.jTFPrecioVenta.getText());
         int exis = Integer.valueOf(this.jTFExistencia.getText());

         String sql = "UPDATE TIENDA..PRODUCTO SET Cod_Categoria = ?, Cod_Subcategoria
= ?, Precio_Compra = ?," +
                  " Precio_Venta = ?, Existencia = ? WHERE Cod_Producto LIKE ?";

         java.sql.PreparedStatement mod = con.conexion.prepareStatement(sql);

         mod.setString(1, codc);
         mod.setString(2, cods);
         mod.setDouble(3, pc);
         mod.setDouble(4, pv);
         mod.setInt(5, exis);
         mod.setString(6, codp);

         mod.executeUpdate();
         con.conexion.close();
      } catch (SQLException ex) {
         JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se puede eliminar datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
puede eliminar datos");
            }
   }

private void showDatos()
   {
      try {
         String CodProducto="";
         String CodSubcategoria="";
         String CodCategoria="";
         Double PrecioCompra=0.0;
         Double PrecioVenta=0.0;
         Integer Existencia=0;

         CONECCION con = new CONECCION();
         String sql = "SELECT * FROM TIENDA..PRODUCTO";
         java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

         del.executeQuery();
         result = del.getResultSet();

int i = 0;
```

```
            while (result.next()) {
                CodProducto = result.getString(1);
                CodSubcategoria = result.getString(2);
                CodCategoria = result.getString(3);
                PrecioCompra = result.getDouble(4);
                PrecioVenta = result.getDouble(5);
                Existencia = result.getInt(6);
   addRow1();
    jTable2.setValueAt(result.getString(1),i, 0);
    jTable2.setValueAt(result.getString(2),i,1);
    jTable2.setValueAt(result.getString(3),i,2);
    jTable2.setValueAt(result.getDouble(4),i,3);
    jTable2.setValueAt(result.getDouble(5),i,4);
    jTable2.setValueAt(result.getInt(6),i,5);
    i++;
        }
            con.conexion.close();
        }
            catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
        }
        catch(NullPointerException ex)
                {
                 JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
                }
    }
}
```

## PANTALLA2

```
package FRAMES;
import CLASES.CONECCION;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class PROD extends javax.swing.JInternalFrame {

    ResultSet result = null;

    public PROD() {
        initComponents();
    }
    private void jBBuscarProductoActionPerformed(java.awt.event.ActionEvent evt) {
     consulta();
}
    private void formPropertyChange(java.beans.PropertyChangeEvent evt) {
     showDatos();
    }
    private void jBSalirActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
    }
```

```
   private void addRow(){
      DefaultTableModel model = (DefaultTableModel)jTable1.getModel();
      Vector fila = new Vector();
      fila.add(null);
      model.addRow(fila);
      jTable1.setModel(model);
   }

private void showDatos()
   {
      try {
         String CodProducto="";
         String CodSubcategoria="";
         String CodCategoria="";
         Double PrecioCompra=0.0;
         Double PrecioVenta=0.0;
         Integer Existencia=0;

         CONECCION con = new CONECCION();
         String sql = "SELECT * FROM TIENDA..PRODUCTO";
         java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

         del.executeQuery();
         result = del.getResultSet();

int i = 0;

         while (result.next()) {
            CodProducto = result.getString(1);
            CodSubcategoria = result.getString(2);
            CodCategoria = result.getString(3);
            PrecioCompra = result.getDouble(4);
            PrecioVenta = result.getDouble(5);
            Existencia = result.getInt(6);

  addRow();

   jTable1.setValueAt(result.getString(1),i, 0);
   jTable1.setValueAt(result.getString(2),i,1);
   jTable1.setValueAt(result.getString(3),i,2);
   jTable1.setValueAt(result.getDouble(4),i,3);
   jTable1.setValueAt(result.getDouble(5),i,4);
   jTable1.setValueAt(result.getInt(6),i,5);
   i++;
      }
         con.conexion.close();
      }
         catch (SQLException ex) {
         JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
```

```
   }

private void consulta()
   {
      try {
          String CodProducto="";
         String CodSubcategoria="";
         String CodCategoria="";
         Double PrecioCompra=0.0;
         Double PrecioVenta=0.0;
         Integer Existencia=0;

         CONECCION con = new CONECCION();
         String cod = JOptionPane.showInputDialog(null,"DIGITE EL CODIGO DEL
PRODUCTO");
         String sql = "SELECT * FROM TIENDA..PRODUCTO WHERE Cod_Producto LIKE ?";
         java.sql.PreparedStatement del = con.conexion.prepareStatement(sql);

         del.setString(1, cod);
         del.executeQuery();
         result = del.getResultSet();

         while (result.next()) {
           CodProducto = result.getString(1);
           CodSubcategoria = result.getString(2);
           CodCategoria = result.getString(3);
           PrecioCompra = result.getDouble(4);
           PrecioVenta = result.getDouble(5);
           Existencia = result.getInt(6);

         this.jTFCodProducto.setText(String.valueOf(CodProducto));
         this.jTFCodSubcategoria.setText(String.valueOf(CodSubcategoria));
         this.jTFCodCategoria.setText(String.valueOf(CodCategoria));
         this.jTFPrecioCompra.setText(String.valueOf(PrecioCompra));
         this.jTFPrecioVenta.setText(String.valueOf(PrecioVenta));
         this.jTFExistencia.setText(String.valueOf(Existencia));

      }
         con.conexion.close();
      }

         catch (SQLException ex) {
         JOptionPane.showMessageDialog(null,ex.getMessage() + " sql Un error ha ocurrido: No
se pueden cargar los datos");
      }
      catch(NullPointerException ex)
            {
             JOptionPane.showMessageDialog(null,ex.getMessage() +"Un error ha ocurrido: No se
pueden cargar los datos");
            }
   }
}
```

# ANEXO IV

**Propuesta 1**

**Componente Detalle**

- ✓ **Procesador** Intel
  Velocidad CPU: Intel: Pentium D 820 2.8 GHz
- ✓ **RAM** 2GB PC2-4200 DDR2
- ✓ **Disco Duro** 160GB SATA (Serial ATA2) con 16 MB en Buffer
- ✓ **Ópticos** CDRW/DVD 8X (combo)
- ✓ **USB** 4 USB 2.0
- ✓ **RED** Ethernet10/100 Mbps
- ✓ **Teclado** PS2 o USB Español
- ✓ **Ratón** PS2 o USB Ópticos, 2 botones con Scrolling
- ✓ **Floppy** 3.5¨ 1.44MB
- ✓ **Video** Tarjeta de Video 128 Mb ATI RADEON X600SE con DVI y TV OUT
- ✓ **Monitor** LCD de 17¨ con fuente externa, res: 1024x768
- ✓ **UPS** Tripp-Lite o APC de 750VA
- ✓ **Estabilizador** Tripp-Lite o APC de 600 VA

**Propuesta 2**

**Componente Detalle**

- ✓ **Procesador** Intel
  Velocidad CPU: Intel: Pentium D 820 1.6 GHz
- ✓ **RAM** 512MB PC2-4200 DDR2
- ✓ **Disco Duro** 40GB SATA (Serial ATA2) con 16 MB en Buffer
- ✓ **Ópticos** CDR
- ✓ **USB** 4 USB 2.0
- ✓ **RED** Ethernet10/100 Mbps
- ✓ **Teclado** PS2 o USB Español
- ✓ **Ratón** PS2 o USB Ópticos, 2 botones con Scrolling
- ✓ **Floppy** 3.5¨ 1.44MB
- ✓ **Video** Tarjeta de Video 128 Mb ATI RADEON X600SE con DVI y TV OUT
- ✓ **Monitor** LCD de 17¨ con fuente externa, res: 1024x768
- ✓ **UPS** Tripp-Lite o APC de 750VA
- ✓ **Estabilizador** Tripp-Lite o APC de 600 VA

**CABLE DE RED TRENZADO (CABLE UTP)**

Es el cable que se utiliza para conexiones de red. Puede ser de varios tipos y categorías, siendo el mas empleado el de categoría 5 (C5), a ser posible blindado. Tiene en su interior 4 pares de cables trenzados y diferenciados por colores (blanco naranja - naranja, blanco verde - verde, blanco azul - azul y blanco marrón - marrón). Esta distribución es el estándar 568-B

**CLAVIJA RJ45**

Es una clavija similar a las de conexión telefónica (RJ11), pero de 11 mm de longitud por 7 mm de grosor, con 8 hilos en vez de 4 ó 6 de las clavijas de teléfono. Aunque tanto los conectores RJ45 como los cables de red tienen 8 hilos, para las funciones de red solo se utilizan los hilos 1, 2, 3 y 6.

**SWITCH**

Sustituto de los hubs, actúan como conmutadores entre ordenadores conectados por cable, permitiendo un mayor y más rápido tráfico de datos entre una o varias redes. Tienen la capacidad de almacenar y discriminar las direcciones de red de los equipos, por lo que se ahorra mucho tráfico de datos inútil.

Para explicarlo de una forma sencilla, el switch, cuando se conecta un ordenador, reconoce su IP, así como la velocidad a la que se conecta, de modo que cuando recibe un paquete de datos para ese ordenador ya sabe a que puerto debe enviarlo, no teniendo que buscar a ver donde esta el ordenador de destino, mientras que el hub cuando recibe un paquete de datos para un ordenador tiene que buscar a ver donde esta ese ordenador. Esto, cuando hablamos de redes de 3 - 4 ordenadores y con poco trafico de datos, prácticamente no tiene importancia, pero imaginaros una red con 20 ordenadores, enviándose datos continuamente entre ellos. Para dar servicio a ordenadores conectados vía WIFI, podemos conectar un Acces Point a uno de los puertos del switch.

## DIRECCIONES IP, PUERTA DE ENLACE Y DNS

La dirección IP es el identificador que tiene nuestro ordenador dentro de la red. Una dirección IP está compuesta por cuatro grupos de entre 1 y 3 dígitos, comprendidos entre los rangos 0 y 255.

**Existen varios tipos de direcciones IP:**

**IP PRIVADA:** Es la IP que le asignamos a nuestro ordenador, no pudiendo estar repetida dentro de nuestra subred.

**MASCARA DE SUBRED:** Es un código numérico que forma parte de la IP y que identifica a la subred, por lo que deberá ser el mismo en todos los ordenadores de la subred. El más habitual es el 255.255.255.0.

**IP PUBLICA:** Es la IP que nos asigna el proveedor de Internet, a fin de identificarnos. Esta IP puede ser dinámica (lo más habitual) o fija.

**PUERTA DE ENLACE:** Es la IP con la que nos conectamos a nuestro router. Normalmente esta IP suele ser 192.168.1.1, si bien puede cambiar.

**DNS:** En realidad es un servicio que se encarga de buscar la dirección IP de una Web a partir del nombre de la misma. Todas las paginas Web tienen una dirección IP, y si la sabemos podemos acceder a ellas mediante esa dirección, pero lo normal es saber su nombre, por lo que necesitamos que la DNS se encargue, a partir de ese nombre, de localizar la dirección IP y conectarnos. La DNS secundaria es la encargada de realizar esta labor si la DNS primaria falla o esta sobrecargada

**DHCP:** es una tecnología para autoconfigurar todas estas direcciones IP. Normalmente esta activada por defecto, tanto en los router como en los acces point.

Las redes profesionales, independientemente del número de ordenadores que las compongan, suelen ser redes montadas con un servidor dedicado. Esto significa que tenemos un ordenador, por lo general muy potente, con sistemas RAID, fuentes de alimentación redundantes y demás sistemas de seguridad, que hace de servidor, el cual no se usa para trabajar, montado normalmente con un sistema operativo especifico para servidores, como el caso del Windows 2003 Server o el Windows 2000 Server, y que es el que contiene tanto los programas como los datos que después van a ejecutar los demás ordenadores. Por seguridad, fiabilidad y rapidez de transmisión de datos, estas redes se suelen montar casi siempre con cableado, utilizándose raramente sistemas WIFI, salvo para dar acceso a la red a algún q otro ordenador portátil. Los estándares de calidad utilizados en los componentes suelen ser superiores a los utilizados en una red domestica (y, por supuesto, bastante mas caros).

En este tipo de redes, a diferencia de las domesticas, es normal que el acceso a Internet se haga a través del servidor, a fin de poder controlar los accesos por parte de los demás usuarios de la red, mediante programas específicos de control de acceso. Suelen trabajar con Dominios, debiéndose dar de alta a los usuarios en el servidor, y no es normal que se conecten puestos de trabajo entre si, sino siempre a través del servidor. En los puestos de trabajo si se suele montar Windows XP o Windows 2000 Workstation, Windows 98, etc. A diferencia de las redes domesticas, en las que es normal gestionar las IP mediante DHCP, este tipo de redes no se suelen configurar mediante DHCP, siendo este otro sistema mas de control y seguridad. Como veréis estoy refiriéndome a redes montadas bajo Windows, pero para grandes redes existen otros sistemas operativos que también se usan, dependiendo esto en gran medida del tipo de negocio de que trate y de con que programas trabajen.

En la actualidad siguen habiendo bastantes grandes redes montadas bajo Novell Netware y bajo Unix y Solaris (estos dos últimos se pueden considerar como los antecesores de Linux).

En redes especiales, donde se precisa transmisión de imagen en tiempo real unido a una gran estabilidad y velocidad, así como distancias grandes, se emplean las redes mediante cable de fibra óptica. El mayor inconveniente de este sistema es su elevadísimo costo.

Hecha esta introducción, vamos a tratar de las redes domesticas. Estas redes no tienen un servidor propiamente dicho, y si bien podemos asignarle a uno de los ordenadores el papel de servidor, sobre todo como contenedor de información, no es necesario hacerlo. Son redes en las que todos los ordenadores ven a todos y en los que la red se utiliza sobre todo para intercambiar ficheros. Por la comodidad que representa, sobre todo al no tener que tender cables, y por tener mayor movilidad los equipos, son redes ideales para sistemas WIFI.

Son redes en las que normalmente todos los ordenadores acceden a Internet (caso de que tengan Internet) a través de un concentrador (normalmente un Router), siendo este el dispositivo de unión entre los equipos que forman la red.

**CONECTAR VARIOS ORDENADORES**

Para conectar varios ordenadores necesitaremos un concentrador, que puede ser un router o un switch, que nos haga de unión entre ellos. En este tipo de conexiones el cable es RECTO, salvo en el caso de algunos router que si usan cables CRUZADOS. Por este sistema podemos poner en red tantos ordenadores como deseemos, dependiendo de la capacidad del switch que tengamos. Además, se pueden conectar varios switch en cascada, estén juntos o separados, por lo que el número de ordenadores que podemos conectar es bastante grande. Por ejemplo, podemos tener 7 ordenadores en una habitación, un cable de red que vaya a otra habitación, otro switch en esta y otros 7 ordenadores conectados.

# ANEXO V

## MANUAL DE USUARIO

## PANTALLA PRINCIPAL

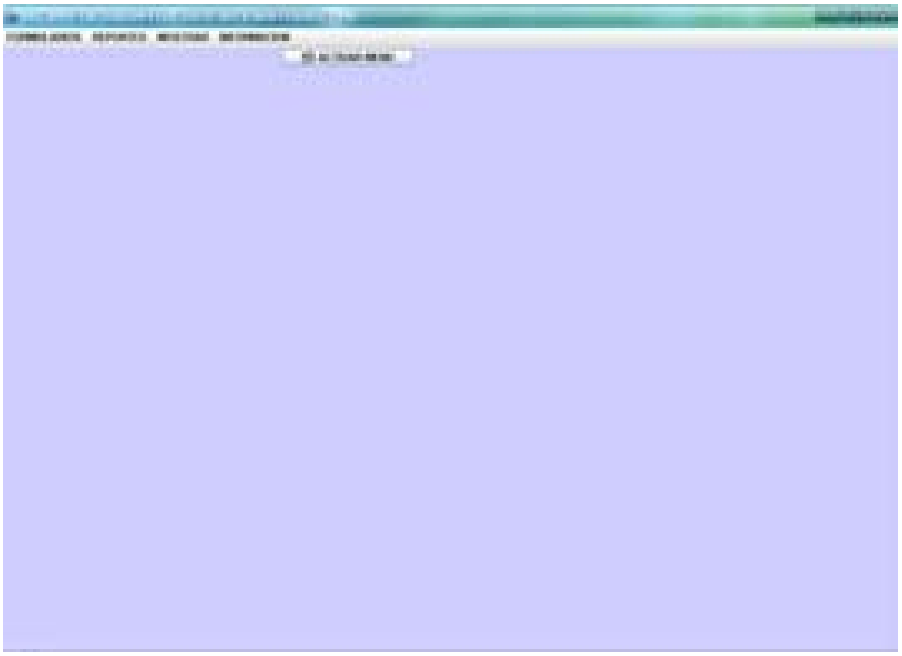


Figura 1

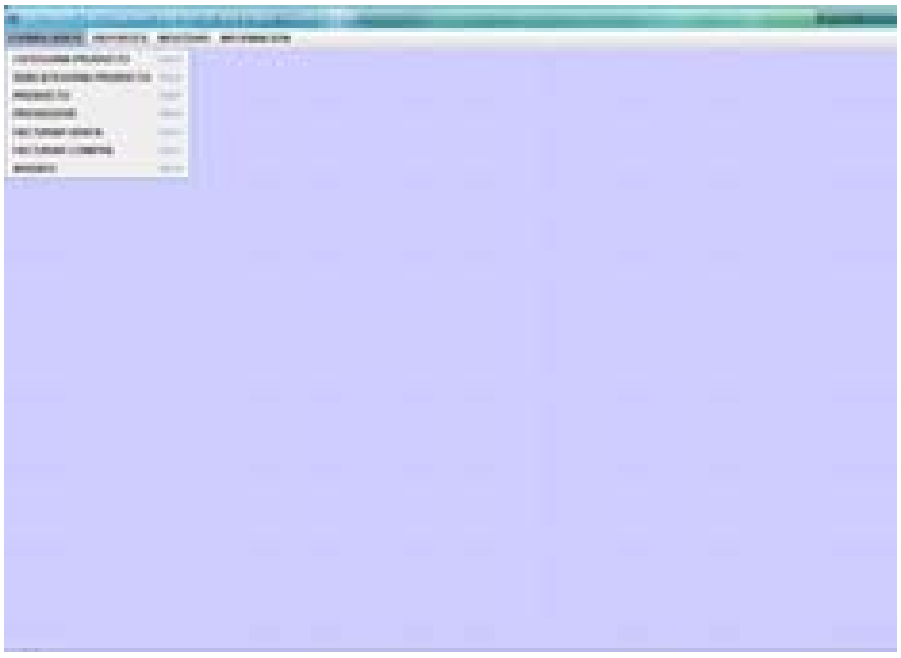


Figura 2

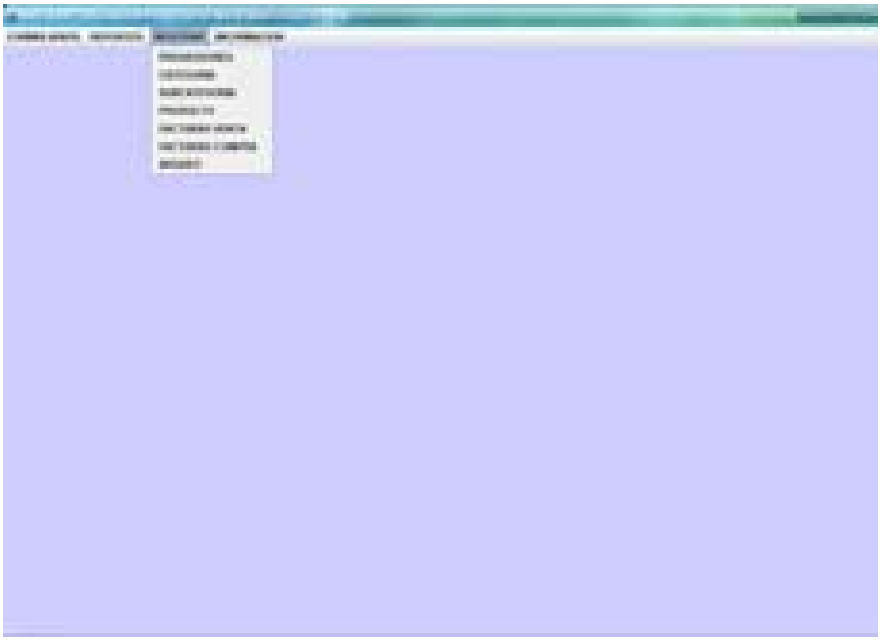


Figura 3

**LOGIN**


Figura 4

**ENTRADA AL SISTEMA**

1. Se selecciona el usuario que se va a utilizar en el ComboBox del usuario (Figura 4)
2. Se digita la clave correspondiente en el TextField de la contraseña.
3. Se presiona el botón aceptar.
4. Estando en la pantalla principal de presiona el botón activar menú (Figura 1).

## CATEGORIA


Figura 5


Figura 6

**REGISTRAR NUEVA CATEGORIA**

1. Se selecciona el menú formularios y después el submenú categoría producto (Figura 2)
2. Se debe de estar dentro de la pantalla categoría producto (Figura 5)
3. Se introduce los datos de la categoría (Cod_Categoria, Categoría)
4. Se presiona el botón agregar fila.
5. Se presiona el botón guardar categoría.
6. Se cierra la ventana

**MODIFICAR CATEGORIA**

1. Se selecciona el menú formularios y después el submenú categoría producto (Figura 2)
2. Se debe de estar dentro de la pantalla categoría producto (Figura 5)
3. Se introduce los datos de la categoría con las modificaciones deseadas (En las cajas de texto).
4. Se presiona el botón modificar categoría.
5. Se cierra la ventana

**ELIMINAR CATEGORIA**

1. Se selecciona el menú formularios y después el submenú categoría producto (Figura 2)
2. Se debe de estar dentro de la pantalla categoría producto (Figura 5)
3. Se digita el código de la categoría que se desea eliminar (Caja de Texto Cod_Categoria).
4. Se presiona el botón eliminar categoría.
5. Se cierra la ventana

**CONSULTAR CATEGORIA**

1. Se selecciona el menú mostrar y después el submenú categoría (Figura 3).
2. Se debe de estar dentro de la pantalla categoría producto (Figura 6)
3. Se presiona el botón buscar categoría.
4. Se introduce el parámetro de consulta (código categoría)
5. Se verifica los datos
6. Se cierra la ventana

## SUBCATEGORIA


Figura 7


Figura 8

**REGISTRAR NUEVA SUBCATEGORIA**

1. Se selecciona el menú formularios y después el submenú subcategoría producto (Figura 2).
2. Se debe de estar dentro de la pantalla subcategoría producto (Figura 7).
3. Se introduce los datos de la subcategoría (Cod_Subcategoria, Cod_Categoria, Subcategoria, Tamaño, Marca).
4. Se presiona el botón agregar fila.
5. Se presiona el botón guardar subcategoría.
6. Se cierra la ventana

**MODIFICAR SUBCATEGORIA**

1. Se selecciona el menú formularios y después el submenú subcategoría producto (Figura 2).
2. Se debe de estar dentro de la pantalla subcategoría (Figura 7).
3. Se introduce los datos de la subcategoría con las modificaciones deseadas (En las cajas de texto).
4. Se presiona el botón modificar subcategoría.
5. Se cierra la ventana

**ELIMINAR SUBCATEGORIA**

1. Se selecciona el menú formularios y después el submenú subcategoría producto (Figura 2).
2. Se debe de estar dentro de la pantalla subcategoría (Figura 7).
3. Se digita el código de la subcategoría que se desea eliminar (Caja de Texto Cod_Subcategoria).
4. Se presiona el botón eliminar subcategoría.
5. Se cierra Ventana

**CONSULTAR SUBCATEGORIA**

1. Se selecciona el menú mostrar y después el submenú subcategoría (Figura 3).
2. Se debe de estar dentro de la pantalla subcategoría producto (Figura 8)
3. Se presiona el botón buscar subcategoría.
4. Se introduce el parámetro de consulta (código subcategoría).
5. Se verifica los datos.
6. Se cierra la ventana.

## PRODUCTO

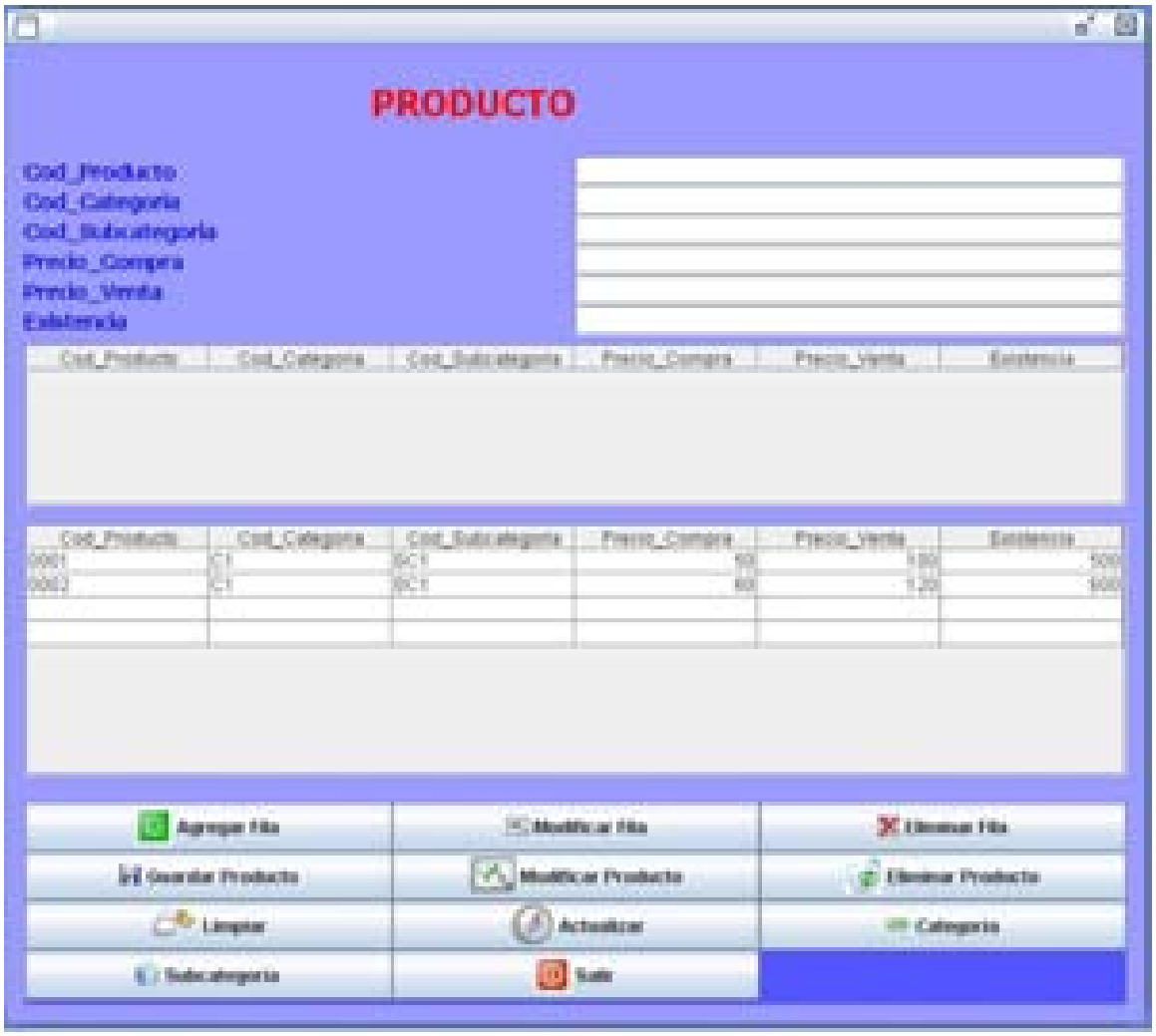



Figura 9

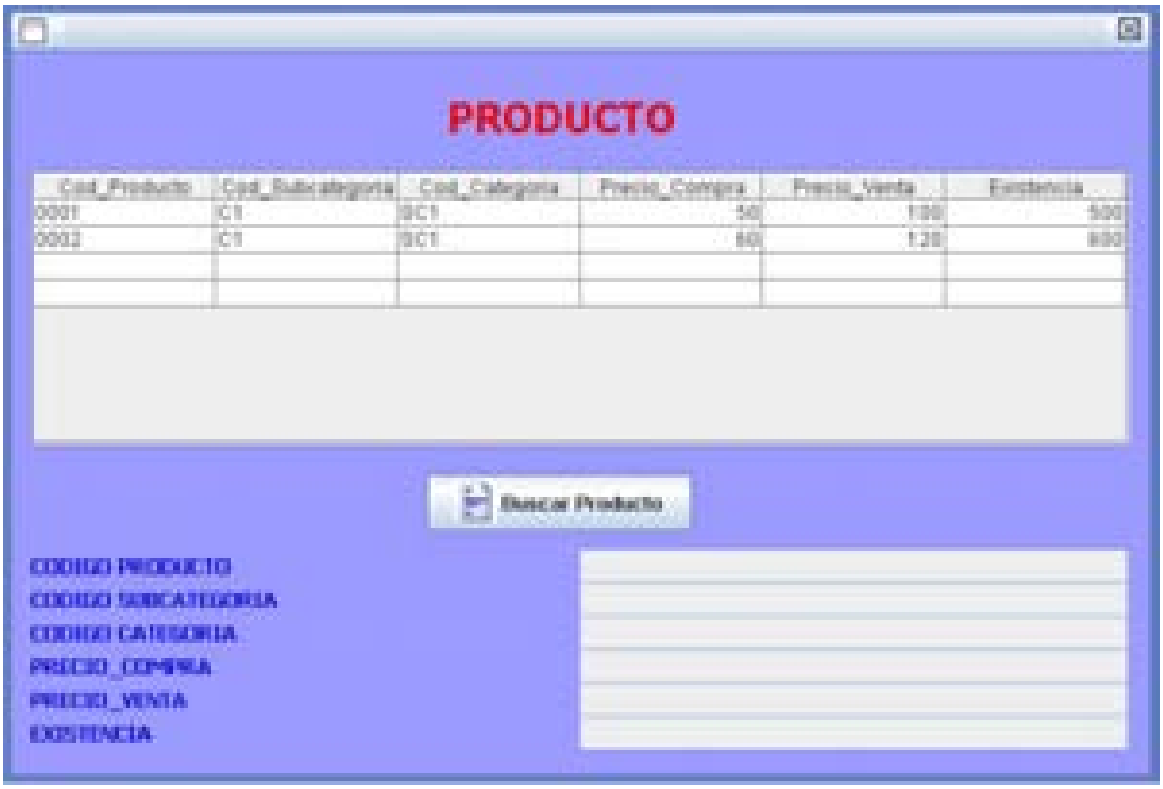



Figura 10

**REGISTRAR NUEVO PRODUCTO**

1. Se selecciona el menú formularios y después el submenú producto (Figura 2).
2. Se debe de estar dentro de la pantalla producto (Figura 9)
3. Se introduce los datos del producto (Código Producto, Código Subcategoria, Código Categoría, Precio_Compra, Precio_Venta, Existencia).
4. Se presiona el botón agregar fila.
5. Se presiona el botón guardar producto.
6. Se cierra la ventana.

**MODIFICAR PRODUCTO**

1. Se selecciona el menú formularios y después el submenú producto (Figura 2).
2. Se debe de estar dentro de la pantalla producto (Figura 9).
3. Se introduce los datos del producto con las modificaciones deseadas (En las cajas de texto).
4. Se presiona el botón modificar producto.
5. Se cierra la ventana

**ELIMINAR PRODUCTO**

1. Se selecciona el menú formularios y después el submenú producto (Figura 2).
2. Se debe de estar dentro de la pantalla producto (Figura 9).
3. Se digita el código del producto que se desea eliminar (Caja de Texto Cod_Producto).
4. Se presiona el botón eliminar producto.
5. Se cierra la ventana

**CONSULTAR PRODUCTO**

1. Se selecciona el menú mostrar y después el submenú producto (Figura 3).
2. Se debe de estar dentro de la pantalla Productos (Figura 10)
3. Se presiona el botón buscar producto.
4. Se introduce el parámetro de consulta (código producto)
5. Se verifica los datos
6. Se cierra la ventana

## PROVEEDOR


Figura 11

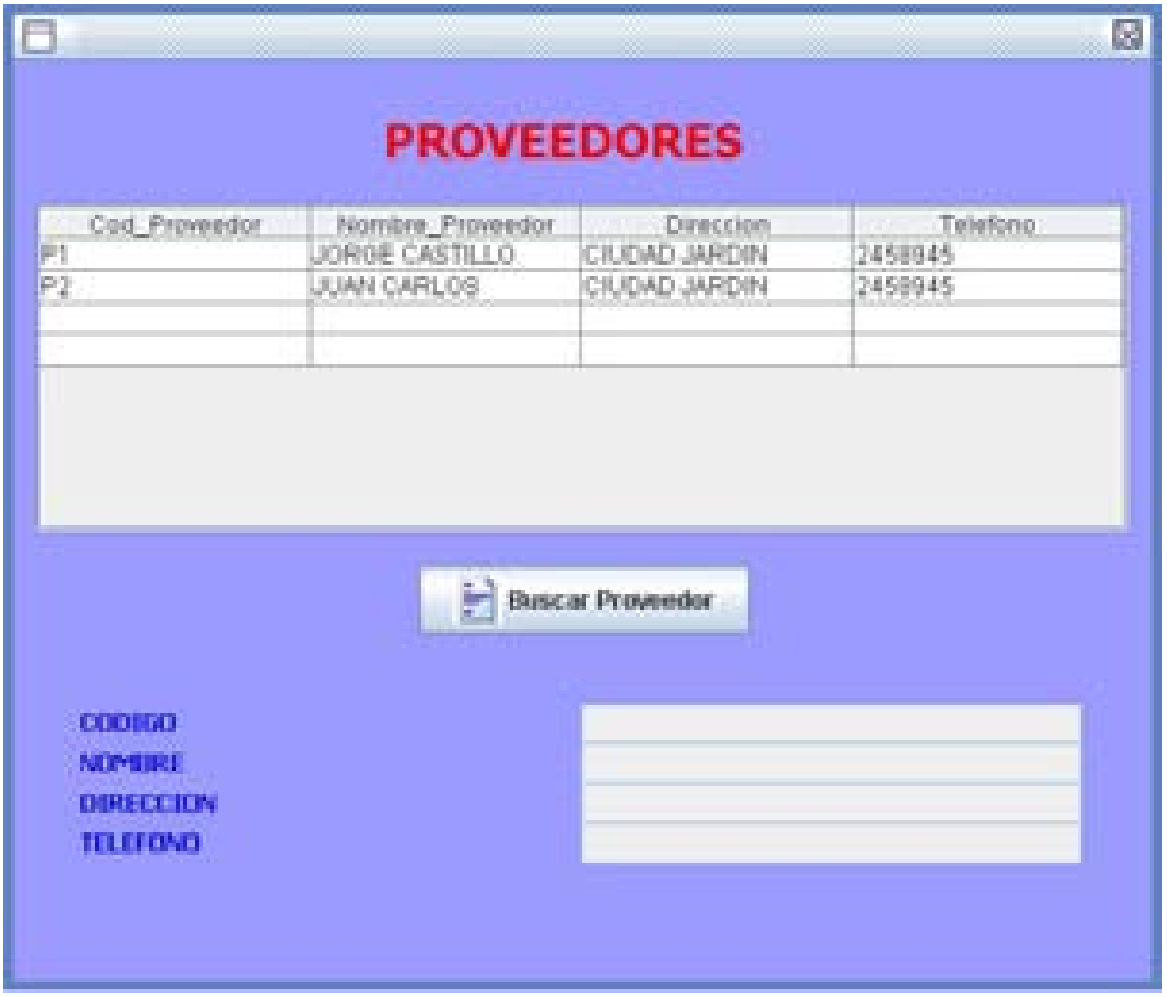



Figura 12

**REGISTRAR NUEVO PROVEEDOR**

1. Se selecciona el menú formularios y después el submenú proveedor (Figura 2).
2. Se debe de estar dentro de la pantalla Proveedor (Figura 11).
3. Se digita los datos del proveedor (Código, Nombre, Dirección, Teléfono).
4. Se presiona el botón agregar fila.
5. Se presiona guardar proveedor.
6. Se cierra la ventana.

**MODIFICAR PROVEEDOR**

1. Se selecciona el menú formularios y después el submenú proveedor (Figura 2).
2. Se debe de estar dentro de la pantalla Proveedor (Figura 11).
3. Se digita los datos del proveedor con las respectivas modificaciones (En las cajas de texto).
4. Se presiona el botón modificar proveedor.
5. Se cierra la ventana.

**ELIMINAR PROVEEDOR**

1. Se selecciona el menú formularios y después el submenú proveedor (Figura 2).
2. Se debe de estar dentro de la pantalla Proveedor (Figura 11).
3. Se digita el código del proveedor a eliminar (Caja de Texto Código).
4. Se presiona el botón eliminar proveedor.
5. Se cierra la ventana.

**CONSULTAR PROVEEDOR**

1. Se selecciona el menú mostrar y después el submenú proveedores (Figura 3).
2. Se debe de estar dentro de la pantalla Proveedor (Figura 12).
3. Se presiona el botón buscar proveedor.
4. Se introduce el parámetro de la consulta (Código Proveedor).
5. Se verifica los datos.
6. Se cierra la ventana.

## ARQUEO


Figura 13


Figura 14

**REGISTRAR NUEVO ARQUEO**

1. Se selecciona el menú formularios y después el submenú arqueos (Figura 2).
2. Se esta dentro de la pantalla Arqueos (Figura 13).
3. Se digita el código y la fecha del arqueo (La fecha se introduce en el orden año, mes y día. Todo separado por un guión. Ej. 2009-01-29).
4. Se presiona el botón Mostrar Facturas para realizar una consulta de las facturas registradas en el día por medio de la fecha de las facturas (Se utiliza el campo Fecha Arqueo como parámetro de consulta).
5. Se presiona el botón Calcular Total Facturas para calcular el total de las facturas reflejadas en la consulta anterior.
6. Se digita el monto en caja chica.
7. Se presiona el botón calcular diferencia (calcula la diferencia entre Total facturas y Caja Chica).
8. Se presiona guardar arqueo.
9. Se cierra la ventana.

**MODIFICAR ARQUEO**

1. Se selecciona el menú formularios y después el submenú arqueos (Figura 2).
2. Se debe de estar dentro de la pantalla Arqueos (Figura 13).
3. Se digita todos los datos del arqueo con las debidas modificaciones (En las cajas de texto).
4. Se presiona el botón modificar arqueo.
5. Se cierra la ventana.

**ELIMINAR ARQUEO**

1. Se selecciona el menú formularios y después el submenú arqueos (Figura 2).
2. Se debe de estar dentro de la pantalla Arqueos (Figura 13).
3. Se digita el código del arqueo que se desea eliminar (Caja de Texto Código Arqueo).
4. Se presiona el botón eliminar arqueo.
5. Se cierra la ventana.

**CONSULTAR ARQUEO**

1. Se selecciona el menú mostrar y después el submenú arqueo (Figura 3).
2. Se debe de estar dentro de la pantalla Arqueos (Figura 14).
3. Se presiona el botón buscar arqueo.
4. Se introduce el parámetro de consulta Correctamente (Código Arqueo).
5. Se verifica los datos.
6. Se cierra la ventana.

## VENTAS


Figura 15


Figura 16

**REGISTRAR NUEVA VENTA**

1. Se selecciona el menú formularios y después el submenú facturar venta (Figura 2).
2. Se debe de estar dentro de la pantalla Facturar Venta (Figura 15).
3. En la parte de facturar venta se digita el código de la factura, la fecha, le descripción y el nombre del cliente.
4. En la parte del detalle de venta se introduce el código de la factura, el código del producto y la cantidad.
5. El sistema verifica la existencia del producto.
6. El sistema rellena el precio automáticamente que se introduce el código del producto y la caja de texto del precio recibe el foco.
7. Se presiona el botón calcular en la parte del detalle de venta.
8. Se presiona el botón calcular total factura en la parte de facturar venta.
9. Se presiona el botón agregar fila en la parte del detalle de venta (Si se pide otro producto en la misma factura se repiten los pasos del 4 al 9).
10. Se digita el monto pagado.
11. Se presiona el botón calcular vuelto.
12. Se presiona el botón agregar fila en la parte de facturar venta.
13. Se presiona el botón Guardar Factura.
14. Se cierra ventana.

**ANULAR FACTURA DE VENTA**

1. Se selecciona el menú formularios y después el submenú facturar venta (Figura 2).
2. Se debe de estar dentro de la pantalla Facturar Venta (Figura 15)
3. Se introduce el código de la factura a eliminar (Caja de Texto Código Factura).
4. Se presiona el botón anular factura.
5. Se cierra la ventana.

**CONSULTAR FACTURA DE VENTA**

1. Se selecciona el menú mostrar y después el submenú facturar venta (Figura 3).
2. Se debe de estar dentro de la pantalla Facturar Venta (Figura 16)
3. Se presiona el botón buscar factura.
4. Se introduce el parámetro de consulta Correctamente (Código de la factura).
5. Se presiona el botón mostrar detalle de factura.
6. Se revisan los datos.
7. Se cierra la ventana.

**COMPRA**


Figura 17


Figura 18

**REGISTRAR NUEVA COMPRA**

1. Se selecciona el menú formularios y después el submenú facturar compra (Figura 2).
2. Se debe de estar dentro de la pantalla Facturar Compra (Figura 17).
3. En la parte de facturar compra se introduce el código de la factura, código proveedor y la fecha.
4. En la parte del detalle de compra se introduce el código de la factura, el código del producto, el producto y la cantidad.
5. El sistema rellena el precio automáticamente que se introduce el código del producto y la caja de texto precio recibe el foco.
6. Se presiona el botón calcular en la parte del detalle de compra.
7. Se presiona el botón calcular total factura en la parte de facturar compra.
8. Se presiona el botón agregar fila en la parte del detalle de compra (Si se pide otro producto en la misma factura se repiten los pasos del 4 al 8).
9. Se presiona el botón agregar fila en la parte de facturar compra.
10. Se presiona el botón Guardar Factura.
11. Se cierra la ventana.

**ANULAR FACTURA DE COMPRA**

1. Se selecciona el menú formularios y después el submenú facturar compra (Figura 2).
2. Se debe de estar dentro de la pantalla Facturar Compra (Figura 17).
3. Se introduce el código de la factura a eliminar (Caja de Texto Código Factura).
4. Se presiona el botón anular factura.
5. Se cierra la ventana.

**CONSULTAR FACTURAS DE COMPRA**

1. Se selecciona el menú mostrar y después el submenú facturar compra (Figura 3).
2. Se debe de estar dentro de la pantalla Facturar Compra (Figura 18)
3. Se presiona el botón buscar factura.
4. Se introduce el parámetro de consulta Correctamente (Código de la factura).
5. Se presiona el botón mostrar detalle de factura.
6. Se revisan los datos.
7. Se cierra la ventana.

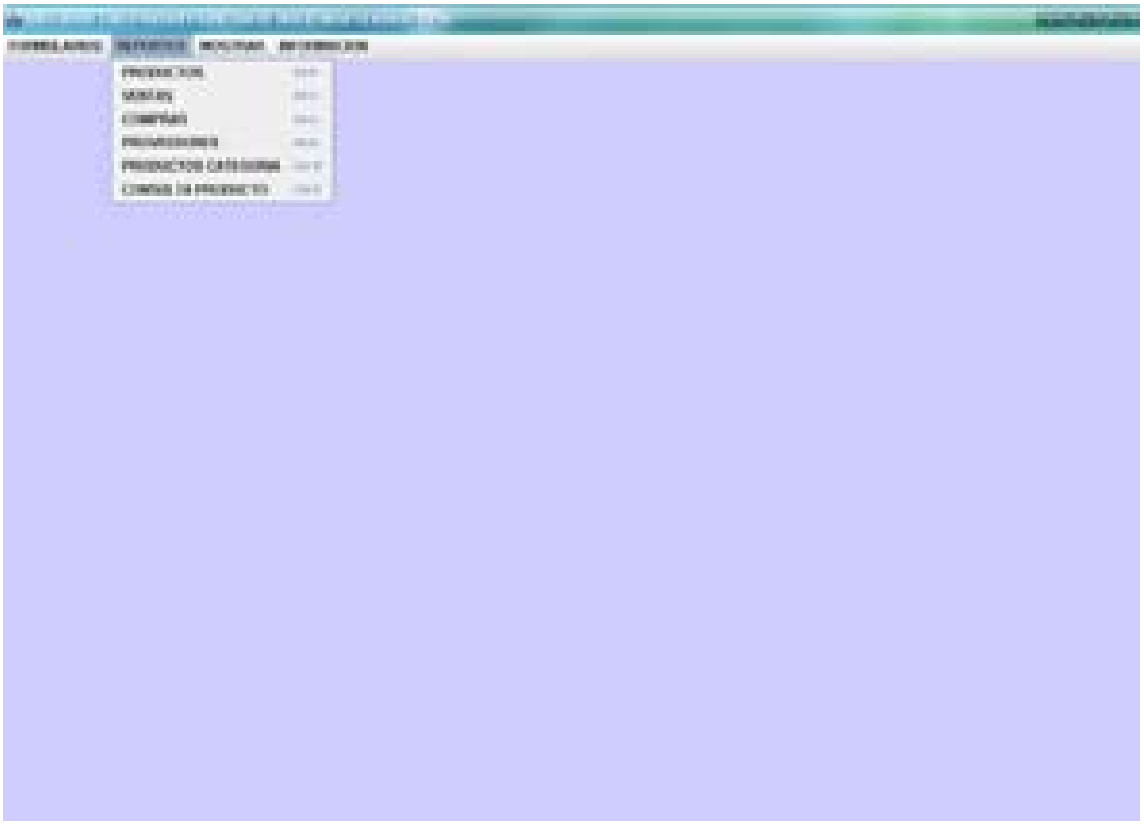


Figura 19

## GENERAR REPORTES

### PRODUCTOS

1. Se selecciona el menú reportes y después el submenú productos (Figura 19).
2. Se revisa los datos reflejado en el reporte.
3. Se manda a imprimir en caso de ser necesario.

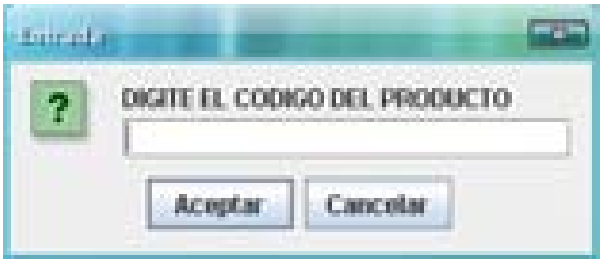



Figura 20

### PRODUCTOS POR MEDIO DE UN PARAMETRO

1. Se selecciona el menú reportes y después el submenú consulta producto (Figura 19).
2. Se introduce el código del producto en la caja de texto (Figura 20)
3. Se presiona el botón aceptar.
4. Se revisa los datos reflejado en el reporte.
5. Se manda a imprimir en caso de ser necesario.

### CATEGORIA

1. Se selecciona el menú reportes y después el submenú productos categoría (Figura 19).
2. Se revisa los datos reflejado en el reporte.
3. Se manda a imprimir en caso de ser necesario.

### PROVEEDORES

1. Se selecciona el menú reportes y después el submenú proveedores (Figura 19).
2. Se revisa los datos reflejado en el reporte.
3. Se manda a imprimir en caso de ser necesario.

### VENTAS

1. Se selecciona el menú reportes y después el submenú ventas (Figura 19).
2. Se revisa los datos reflejado en el reporte.
3. Se manda a imprimir en caso de ser necesario.

### COMPRAS

1. Se selecciona el menú reportes y después el submenú compras (Figura 19).
2. Se revisa los datos reflejado en el reporte.
3. Se manda a imprimir en caso de ser necesario.