CAPITULO I

EL PROBLEMA

1.1 Tema

"Sistema de control de productos y servicios de la Ferretería Bolivariana para evitar pérdidas económicas"

1.2 Planteamiento del Problema

1.2.1 Contextualización

En este año las empresas han sufrido grandes problemas económicos debido al nuevo modelo económico implantado en nuestro país, por lo que en muchas de ellas las ventas han disminuido notablemente, incluso varias han tenido que cerrar sus puertas aumentándose el índice de desempleo en el Ecuador.

Los datos demuestran que la situación económica en uno de los principales centros tecnológicos del mundo comienza a ser preocupante, y de aquí que afecte tanto el desarrollo tecnológico como el empleo el cual ha bajado un 1,3% a medida que más compañías tecnológicas reducen puestos de trabajo. Aquí no puede hablarse de despidos masivos como en otros sectores, pero firmas como Microsoft o Yahoo han recortado ya miles de empleos y otras como Google han dejado de contratar.

En Tungurahua se ha hecho caso omiso a la utilización de sistemas computacionales en las pequeñas y medianas empresas debido a la falta de recursos económicos, esto implica no tener un sistema de control y ha provocado el mal manejo de la información trayendo como consecuencia la derrota de muchas empresas.

En la ciudad de Ambato existe muy poca acogida al avance tecnológico y la población prefiere no cambiar su manera de trabajar, ni aprender a manejar un ordenador con un sistema implementado, debido a esto la mayoría de empresas pequeñas y medianas llevan su información de forma manual y no permiten el cambio a la sistematización.

1.2.2 Análisis Crítico

Las pérdidas económicas por falta de control en existencias de los productos, compras y ventas de la empresa es una de las causas más graves por lo que se ha llegado a pensar en la elaboración de este proyecto.

La Ferretería Bolivariana lleva la información de sus proveedores y clientes en copias de fichas lo cual no da una buena imagen a la empresa.

Esta empresa factura de forma manual lo cual implica pérdida de tiempo al hacer los cálculos matemáticos, y por ende insatisfacción de los clientes en la espera de su factura.

La pérdida de ventas en la empresa se debe a la falta de información oportuna ya que no se cuenta con un detalle de productos y sus respectivos precios de una manera automática.

Por esta razón la propietaria de la Ferretería Bolivariana ha buscado ayuda para no seguir teniendo pérdidas económicas.

1.2.3 Prognosis

Si la empresa no ejecuta la solución a este problema mediante la implementación del sistema de control tendrá graves consecuencias como la derrota de su empresa por falta de control y un mal servicio al cliente.

1.3 Formulación del Problema

¿Cómo incidiría el desarrollo e implementación de un sistema informático de control de productos y servicios en la Ferretería Bolivariana?

1.4 Preguntas Directrices

¿Qué procedimientos de compras, proveedores e inventarios de la empresa debemos conocer?

¿Qué factores son importantes para el diseño y desarrollo del sistema?

¿Cómo se realizará la implementación del sistema?

¿Qué implica la administración del sistema?

¿Qué herramientas de desarrollo permitirán optimizar los procesos del sistema?

1.5 Delimitación del Problema

El proyecto "Sistema de control de productos y servicios de la Ferretería Bolivariana para evitar pérdidas económicas", será desarrollado para dicha empresa ubicada en la Av. Bolivariana 5-12 y Seimour del cantón Ambato. El presente trabajo se pretende desarrollar en el periodo Abril-Octubre del 2009.

1.6 Justificación

La empresa Ferretería Bolivariana tiene como finalidad atender los clientes locales y de la provincia con productos de ferretería y materiales de construcción.

Junto con el crecimiento de la empresa ha crecido también la necesidad de mejorar el proceso en la atención a los clientes, utilizando un método actual, rápido y seguro como lo es el usado sistema informático que realice la función que hoy en día la propietaria lo hace manualmente.

El problema que se solucionara es la falta de control de productos y servicios mediante un modulo de control de existencias basado en las facturas de compra y venta, así como también la mala organización de la información ya que el sistema tendrá la información ordenada y al alcance de las personas que lo necesitan.

Con la implantación de este Sistema Automatizado se contribuye a tener un orden en el esquema funcional de la empresa así como en el desenvolvimiento de la misma.

Es importante también debido a que la competitividad que la empresa tendrá frente a otras permitirá su superación a nivel Cantonal, y así ser conocida y divulgada de mejor manera.

Los beneficios que brindará la implantación del sistema contable son muy factibles, ya que analizando el problema que viene atravesando la empresa se ha determinado que es menester aplicar este sistema, enmarcándonos con reglamentos y leyes que rigen a la empresa en mención.

Por la importancia que tiene aplicar los sistemas informáticos en microempresas que carecen de ellos para así promulgarla el avance tecnológico en su entorno.

Por cumplir con una obligación que tenemos como estudiantes universitarios a contribuir con el desarrollo de la investigación en áreas aplicables como es sistemas informáticos para zonas que desconocen de lo beneficioso de la tecnología.

Por los beneficios que el socio tendrá al no tener que esperar para que le realicen su factura manualmente así como el tiempo que se ahorrara la propietaria al no buscar o inventarse precios.

1.6.1 Utilidad Teórica

Es de mucha importancia que se implante un sistema de automatización en la "Ferretería Bolivariana", ya que los procesos manuales dentro de una empresa en la actualidad ya no se usan, es indispensable que la sociedad se vaya empapando de los beneficios que la tecnología trae a las empresas y al diario vivir de la personas.

1.6.2 Utilidad Metodológica

El sistema será desarrollado utilizando el lenguaje de programación Visual Studio 2005 en C#, ya que cuenta con todas las herramientas necesarias para elaborar el sistema.

1.7 Objetivos de la investigación

1.7.1 Objetivo General

"Desarrollar e Implementar un sistema de control de productos y servicios de la Ferretería Bolivariana para evitar pérdidas económicas"

1.7.2 Objetivos Específicos

- Analizar los procedimientos de compras, proveedores e inventarios de la empresa para determinar el mecanismo adecuado para el manejo automático de dichos procesos.
- Crear procesos que permitan la correcta administración del sistema actual
- Plantear una propuesta informática de solución utilizando base de datos relacional y tecnología, para el control de productos y servicios en la Ferretería Bolivariana.

CAPITULO II

MARCO TEORICO

2.1 Antecedentes Investigativos

Una vez revisados los archivos de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial se ha encontrado que en la biblioteca reposa una investigación con el tema:

CONTROL DE COMPRAS, PROVEEDORES E INVENTARIOS BAJO UNA ARQUITECTURA CLIENTE-SERVIDOR, realizado por ORTIZ MARIANELA y VARELA VERONICA, donde se maneja el control de entrada y salida de la mercadería, también aborda la temática que permite tener una arquitectura cliente servidor para mayor eficiencia de la empresa en una sucursal.

2.2 Fundamentación

Se trabajará con una metodología dialéctica que permitirá analizar los hechos y fenómenos de automatización reales, y por la misma razón susceptibles de ser medidos, valorados, cuantificados, demostrados y comprobados evitando de esta manera caer en el campo del idealismo y más bien ofrecer respuestas a los problemas derivados de estos permitiendo de esta manera ser completamente objetivo y realista.

El ambiente empresarial, requiere que la información esté siempre disponible, a toda hora y en todo lugar, lo cual nos indica que el tener una base de datos es totalmente necesaria, además que se pueda tener acceso desde otro computador, en este caso el o los usuarios que manipulen el sistema, en el cual el bodeguero actualizará los productos que entren y salgan de bodega, la vendedora realizará las ventas directamente con el cliente, en el cual debe notificarse al repartidor y también actualizar el sistema en bodega, haciendo el proceso denominado cálculo de existencias de productos.

2.2.1 Fundamentación Legal

La microempresa "Ferretería Bolivariana", dentro de su constitución legal cuentas con el siguiente documento que lo certifica como una entidad plenamente estructurada:

RUC (Registro Único de Contribuyentes) que consta en el registro del servicio de Rentas Internas.

2.2.2 Fundamentación Teórica

Para la elaboración del proyecto se debe conocer previamente los siguientes conceptos:

2.2.2.1 La contabilidad como ciencia aplicada

Es una ciencia aplicada por que toma como estudio los problemas y fenómenos económicos y/o financieros que se presenta en el desarrollo o funcionamiento de toda entidad o institución, y buscará mediante un previo análisis soluciones a los problemas originados, gracias a sus técnicas, métodos y principios dará una información con mayor objetividad, para que así el funcionamiento del ente sea normal.[1]

[1] http://www.monografias.com/trabajos12/evintven/evitven .shtml

2.2.2.2 Compras

Compras es un proceso en el que participan el solicitante que formula el requerimiento de un bien tanto de patrimonio como un bien para el consumo en el proceso de su actividad dentro de la institución

2.2.2.3 Proveedores

Son proveedores aquellas personas físicas o jurídicas que surten a la empresa de existencias (mercaderías, materia prima, envases, etc.), que posteriormente ésta venderá, transformará o elaborará, Así, en un almacén de materiales de construcción, los proveedores serán aquellas empresas que le suministren:

Cemento, pintura, vigas, azulejos, pernos, clavos, material eléctrico, etc.

La compra de productos se puede pagar al contado. Contablemente, este tipo de operaciones se registrara mediante un asiento como el siguiente:

Concepto	Debe	Haber
Compras de Existencias (60)		
Tesorería		

Sin embargo, en el tráfico mercantil, es muy corriente que en esta clase de transacciones se aplace el pago a 30, 60 o 90 días. De este modo, el adquirente no paga las existencias en el momento de la compra, sino que obtiene financiación de los proveedores, disponiendo de mayor liquidez durante el periodo que aplaza el pago, que empleará en otros usos, incluso, en ese intervalo de tiempo, probablemente venda las mercancías y con los recursos que obtenga abone a los proveedores.

2.2.2.4 Manejo de Inventario

Inventarios son bienes tangibles que se tienen para la venta en el curso ordinario del negocio o para ser consumidos en la producción de bienes o servicios para su posterior comercialización. Los inventarios comprenden, además de las materias primas, productos en proceso y terminados o mercancías para la venta, los materiales, repuestos y accesorios para ser consumidos en la producción de bienes fabricados para la venta o en la prestación de servicios, empaques y envases y los inventarios en tránsito.

La base de toda empresa comercial es la compra y venta de bienes y servicios; de aquí la importancia del manejo del inventario por parte de la misma. Este manejo contable permitirá a la empresa mantener el control oportunamente, así como también conocer al final del periodo contable un estado confiable de la situación económica de la empresa.[2]

Funciones:

- Eliminación de irregularidades en la oferta
- Compra o producción en lotes o tandas
- Permitir a la organización manejar materiales perecederos
- Almacenamiento de mano de obra

2.2.2.5 Cuentas por Pagar

Activo circulante exigible, comprende las cantidades por cobrar en cuenta abierta a los deudores por operaciones propias del negocio, las cuentas por cobrar a los accionistas, funcionarios o empleados.

[2] http://www.um.es/gtiweb/jgomez/bibgen/intranet/02conceptob.PDF

2.2.2.6 Arquitectura Cliente Servidor

Esta arquitectura consiste básicamente en que un programa (el cliente) realiza peticiones a otro programa (el servidor) que le da la respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajoso en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debido a la centralización de la gestión de la información y la separación de las responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesario un solo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, servidores de correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

2.2.2.6.1 Ventajas

Centralización del Control: Los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).

Escalabilidad: Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado o separado en cualquier momento, o se puede añadir nuevos nodos a la red (clientes y/o servidores).

Fácil Mantenimiento: Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectaran mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

Existen tecnologías suficientemente desarrolladas, diseñadas para el paradigma de cliente-servidor que aseguran las transacciones, la amigabilidad del interfaz, y la facilidad del empleo.

2.2.2.7 Base de Datos.

A lo largo de la historia el término Bases de Datos ha tenido múltiples interpretaciones, desde los años 60´s cuando se acuñó el término. Algunas definiciones comunes son:

- "Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados" (Martin, 1975).

- "Colección o depósito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular. Una base de datos es, también, un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones" (Conferencia des Statisticiens Européens, 1977)".

- "Conjunto de datos de la empresa memorizado por un ordenador, que es utilizado por numerosas personas y cuya organización está regida por un modelo de datos" (Flory, 1982).

- "Conjunto estructurado de datos registrados sobre soportes accesibles por ordenador para satisfacer simultáneamente a varios usuarios de forma selectiva y en tiempo oportuno" (Delobel, 1982).

- "Colección no redundante de datos compartibles entre diferentes sistemas de aplicación" (Howe, 1983).

- "Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender toda las necesidades de los diferentes usuarios". (Deen, 1985)

- "Conjunto de ficheros maestros, organizados y administrados de una manera flexible de modo que los ficheros puedan ser fácilmente adaptados a nuevas tareas imprevisibles" (Frank, 1988).

- "Colección de datos interrelacionados" (Emasri y Navathe, 1989).

Una definición generalizada pudiera ser:

- “Colección o depósito de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos

de actualización y recuperación, comunes y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos"

2.2.2.7.1 Objetivos de las bases de datos

El hecho de emplear sistemas de bases de datos, busca unos objetivos específicos que podemos enumerar de la siguiente forma:

Los datos podrán utilizarse de múltiples maneras.

Se protegerá la inversión intelectual.

- Bajo Costo.
- Menor proliferación de datos.
- Desempeño.
- Claridad.
- Facilidad de uso.
- Flexibilidad.
- Rápida atención de interrogantes no previstos.
- Facilidad para el cambio.
- Precisión y coherencia.
- Reserva.
- Protección contra pérdida o daño.
- Disponibilidad.

Para lograr el cumplimiento de los anteriores objetivos, se tienen algunos secundarios como los siguientes:

- Independencia física de los datos.
- Independencia lógica de los datos.
- Redundancia controlada.
- Adecuada rapidez de acceso.

- Adecuada rapidez de exploración.
- Normalización de los datos dentro de un organismo.
- Diccionario de datos.
- Interface de alto nivel con los programadores.
- Lenguaje de usuario final.
- Controles de integridad.
- Fácil recuperación en caso de fallo.
- Afinación.
- Ayudas para el diseño y la supervisión.
- Migración reorganización automática.

2.2.2.7.2 Ventajas del Empleo de Bases de Datos

De los Datos

- Independencia de éstos respecto de los tratamientos y viceversa.
- Mejor disponibilidad de los mismos.
- Mayor eficiencia en la recogida, codificación y entrada al sistema.

De Los Resultados

- Mayor coherencia
- Mayor valor informativo
- Mejor y más normalizada documentación de la información

De Los Usuarios

- Acceso más rápido y sencillo de los usuarios finales.
- Más facilidades para compartir los datos por el conjunto de los usuarios.
- Mayor flexibilidad para atender a demandas cambiantes.

2.2.2.7.3 Inconvenientes del empleo de Bases de Datos

De la Implantación

Costosa en equipo (físico y lógico)

Ausencia de Estándares

Larga y difícil puesta en marcha

Rentabilidad a mediano plazo[3]

2.2.2.8 SQL Server

SQL Server en una herramienta completa destinada a la administración y gestión de base de datos de pequeñas y grandes empresas.

2.2.2.8.1 Lenguajes que maneja SQL SERVER

1. Lenguaje de descripción de datos LDD: Permite describir y definir todos los objetos o esquemas que van en la base de datos
2. Lenguaje de manipulación de datos LMD: Permite definir operaciones entre tablas como consultas y actualizaciones
3. Lenguaje de control de datos LCD: Permite determinar los diferentes accesos a los usuarios: Control sobre la base de datos, Control sobre las tablas.

2.2.2.8.2 Estructura del SQL SERVER

1. Base de datos: Es la estructura fundamental que está compuesta de varios elementos u objetos.
2. Tablas: Conjunto de filas y columnas que contienen la información específica.

[3] http://www.google.com.ec/search?hl=es&q=define:Registro&sa=X&oi=gloss

3. Vistas: Son tablas virtuales definidas sobre tablas creadas que permiten manipular los datos y las relaciones entre las tablas.
4. Usuarios: Conjunto de accesos autorizados con sus respectivos permisos para la manipulación de los datos.
5. Índices: Permite aumentar el rendimiento de la base del sistema y asegurar la integridad de los datos. A cada tabla se le asocia automáticamente una tabla índice que contiene la posición del registro según la columna especificada como índice.

2.2.2.8.3 Uso de la Memoria en SQL

- La configuración por defecto de SQL Server en lo que tiene que ver al espacio de memoria está dividida en dos espacios:
- El cache de procedimientos.- Contiene los procedimientos almacenados y usados frecuentemente
- El cache de datos.- Contiene las tablas e índices usados frecuentemente.
- Estos dos espacios son administrados automáticamente por el lenguaje según la configuración del usuario.
- Se puede determinar el uso de la memoria con el comando: DBCC MEMUSAGE. Esta sentencia indica la cantidad de memoria asignada al servidor SQL.

2.2.2.8.4 Ventajas del SQL

- Contiene un completo conjunto de herramientas necesarias para la administración y gestión de base de datos empresariales.
- Compatible con todas las aplicaciones de gestión que se encuentran en el mercado.
- Posee una alta escalabilidad adaptándose a las necesidades de la empresa.
- Independiente de cualquier lenguaje de desarrollo.
- Está orientado al desarrollo, lo que le permite un fácil enlace con lenguajes visuales como: Visual Basic, Visual C++, Visual J++, .Net, etc.
- Orientado a entornos de trabajo en red como: Internet o Intranet.

- Es una plataforma de desarrollo abierta que integra la tecnología Active X.
- Permite generación de código HTML de forma automática.

2.2.2.8.5 Herramientas De SQL Server

Entre el conjunto de herramientas que integra podemos citar:

- SQL Enterprice Manager: Consola de gestión visual en ambiente windows.
- SQL Executive: planificador de trabajos y monitor de gestión de servidores distribuidos.
- Scripts Visual Basic a través de SQL Distributed Management Objects (SQL-DMO) basados en OLE.
- DBA Assistant: Para mantenimiento automático rutinario.
- SQL Trace, para el monitoreo de consultas cliente-servidor.

2.2.2.8.6 Características Del SQL

- Permite realizar copias de seguridad online desatendidas.
- Recuperación automática de fallos de dispositivos.
- Contextos de usuarios protegidos, que permiten el aislamiento de fallos.
- Recuperación y restauración de base de datos o transacción logs en un intervalo de tiempo.
- Tolerancia a fallos de servidor, permitiendo failover automático a un servidor de backup o en espera.
- Disparador de eventos (triggers) antes y después de conexiones.
- Procedimientos almacenados extendidos utilizando C/C++.

2.2.2.9 Seguridades

- Posee un único ID de login tanto para red como para la DB para mejorar la seguridad y la facilitar la administración.

- Password y encriptación de datos en red para mejorar la seguridad.
- Encriptación de procedimientos almacenados para la integridad y seguridad del código de aplicación.
- Estándar ODBC Nivel 2 totalmente soportado por API nativa.

2.2.2.10 Sistema de gestión de base de datos.

Es una interfaz software entre la base de datos y el usuario. Un sistema de gestión de base de datos maneta las solicitudes de los usuarios para realizar acciones de base de datos y permite cumplir con los requisitos de control de la seguridad e integridad de los datos.

2.2.2.11 Metodologías Utilizadas

En el desarrollo de un sistema automatizado es necesario utilizar uno de los enfoques más generalizados para facilitar la planificación, dirección, realización, y control del mismo, esto lo podemos lograr mediante el uso de las metodologías.

La metodología que vamos a utilizar en el sistema va hacer ADESA (Análisis y Diseño de Sistemas Automatizados) que ofrece una serie de herramientas como el diagrama entidad-relación, diagrama de flujo de datos, diccionario de datos, etc.

Además se incluirá otras herramientas de análisis y diseño orientadas a la arquitectura cliente-servidor entre los que podemos mencionar las reglas de negocio y su ubicación en dicho modelo, distribución u enlazado de componentes de software, diseño de base de datos, etc.

2.2.2.12 Establecimiento de Reglas de Negocio

Toda aplicación trata de reflejar parte del funcionamiento del mundo real, para automatizar tareas que de otro modo sería llevado a cabo de manera más eficiente, o bien no podían realizarse. Para ello es necesario que cada aplicación refleje las restricciones que

existan en el negocio dado, de modo que nunca sea posible llevar a cabo acciones no validas. A las reglas que debe seguir la aplicación para garantizar esto se las llama reglas de negocio.

2.2.2.13 ¿Qué procedimientos de compras, proveedores e inventarios de la empresa debemos conocer?

Se debe conocer la gama de productos y servicios que la empresa oferta, así también todos sus proveedores y clientes para poder determinar el funcionamiento de la organización.

2.2.2.14 ¿Qué factores son importantes para el diseño y desarrollo del sistema?

Hay que tener encuentra que para el desarrollo de sistemas informáticos es muy importante la metodología que se utilizara incluyendo en ésta el mayor avance tecnológico, el soporte de la base de datos y un diseño de interfaz amigable.

Para esto se desarrollara las siguientes partes del sistema:

- Diseño de la interfaz
- Barra de Menús
- Elaboración de Pedidos
- Elaboración de Facturas
- Realizar Pagos y Cobros
- Transacciones e Inventarios
- Sistema de Ayuda
- Sistema de Acceso y permiso de usuarios

2.2.2.15 Implantación de un sistema automatizado.

Un sistema automatizado es un sistema informático, para el proceso y organización de información de forma que proporcione varios niveles de gestión dentro de una

organización con información segura y actualizada necesaria para actividades de supervisión, seguimiento, toma de decisiones y solución de problemas.

En nuestro caso se implementará el sistema en versión prueba durante un mes, en el cual se depurara los últimos errores del sistema, después de esto inmediatamente se implantara ele sistema beta del sistema.

2.2.2.16 ¿Qué flexibilidad tiene el sistema?

El sistema puede ser modificado por cualquier profesional ya que cuenta con una documentación minuciosa para facilitar la administración del mismo.

2.3 Variables

2.3.1 Variable Independiente

Sistema de control de productos y servicios

2.3.2 Variable Dependiente

Pérdidas económicas en la Ferretería Bolivariana

2.4 Hipótesis

La implementación de un sistema de control de productos y servicios en la Ferretería Bolivariana evitará pérdidas económicas producidas por el mal manejo de la información.

CAPITULO III

METODOLOGIA

3.1 Enfoque

3.1.1 Paradigma Cualitativo

La investigación que se realizará tiene un enfoque cualitativo ya que es contextualizado, pone énfasis en buscar menos la generalización ya que se tiene como único propósito solucionar problemas de una empresa, discute la validez del conocimiento utilizando varias técnicas como la observación.

3.1.2 Paradigma Cuantitativo

Esta investigación es cuantitativa porque se basa en un positivismo lógico que busca las causas, la explicación de los hechos que originan al problema, y como estos afectan al desenvolvimiento de la institución.

Al definir los objetivos estos se orientaran a la obtención de resultados y además deberán estar orientados a la solución del problema de la institución en un contexto enmarcado en lo tangible, estable y sostenible.

3.2 Modalidad Básica de la Investigación

El presente trabajo que se propone desarrollar bajo los normativos de una investigación de campo, es decir se tendrá la oportunidad de investigar en el lugar de los hechos, con la finalidad de recolectar y registrar ordenadamente la información de todas las transacciones ejecutadas por los empleados de la empresa.

Para la solución del problema de Investigación se partirá con una investigación de tipo bibliográfica que permitirá conseguir la base científica para la solución de la problemática indicada que a pesar de presentar características muy limitadas será minuciosamente estudiada. El proyecto está dentro de la factibilidad porque me permite solucionar el problema suscitado en base teórica.

3.3 Nivel o Tipo de Investigación

En la investigación se partirá del nivel exploratorio, ya que nos permite conocer y contextualizar el problema; el nivel de tipo descriptivo, ya que facilita la identificación de las variables, el análisis crítico de la situación, también permite enlistar lo que se observo en la empresa en la fase de investigación de campo, recogiendo datos reales de los usuarios de la Ferretería Bolivariana; el nivel correlacional ayuda a establecer relaciones entre las causas y los efectos, la variable independiente y al variable dependiente.

Es necesario puntualizar que el trabajo presentara características de una investigación combinada porque será necesario utilizar investigación: exploratoria, de campo, bibliográfica y documentada para familiarizarse o ponerse en contacto con la realidad del problema que se va a estudiar, se empleara las técnicas de observación, entrevista y documental para la recolección de información y obtención de datos.

3.4 Población y Muestra

3.4.1 Población

El universo de investigación está representado por la "Ferretería Bolivariana" y el trabajo se desarrolla con la población en su totalidad, que consta de cinco personas, y al ser esta pequeña no amerita aplicar el método de muestreo.

3.5 Operacionalización de las variables

Variable Independiente: Sistema de Control de Productos y Servicios

Conceptualización	Categoría	Indicadores	Items	Técnica e Instrumento
Sistema de control de productos y servicios	Sistema	Seguridad	Procesos	Observación bibliográfica técnica
	Control	Rapidez	Calidad Versatilidad	
	Productos			

	Servicios	Plataforma	Seguridad Informática	

Tabla 1.Operacionalizacion de la variable independiente

Variable Dependiente: Pérdidas económicas en la Ferretería Bolivariana

Conceptualización	Categoría	Indicadores	Items	Técnica e Instrumento
		Cantidad de productos	¿Qué cantidad de productos se maneja en bodega mensualmente?	Observación bibliográfica técnica
		Replica	¿La empresa tiene o tendrá sucursales?	
		Implantación	¿Qué porcentaje de riesgo tiene el sistema?	

		Productos	¿Tiene su empresa clasificados los productos de alguna manera?	
		Proveedores	¿Con cuántos proveedores cuenta la empresa?	
		Clientes	¿Qué número de clientes existe en la actualidad en su negocio?	
		Servicios	¿Clases de servicios que presta y costo de cada servicio?	
		Perdidas en Bodega	Numero de pérdidas económicas Numero de productos perdidos	

Tabla 2. Operacionalizacion de la variable dependiente

3.6 Recopilación de la Información

Para la recolección de información de la investigación, se realizará una entrevista a la muestra seleccionada.

3.6.1 Plan para la Recolección de la Información

Para la recolección eficaz de la información de campo se recurrirá al diseño y la elaboración de un cuestionario que contendrá lo medular para desarrollar el sistema.

Las personas que van ser encuestadas para la recolección de datos serán la propietaria de la Ferretería Bolivariana y los empleados porque son parte de la población a quienes afecta el problema esto se realizará dentro del mes Abril.

3.6.2 Plan para el Procesamiento de la Información

Luego de haber aplicado las encuestas a las personas respectivas se procederá con la revisión del cuestionario, para con ello realizar estudios estadísticos y una respectiva tabulación para facilitar el entendimiento de la información recolectada. Este procesamiento de la información lo realizare a través de Memorias Técnicas.

CAPITULO IV

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS.

4.1 Recopilación de Información

Se escogió la Encuesta como técnica para recolectar información acerca del manejo de información de la empresa y sus actividades y está dirigida a la población de la Ferretería Bolivariana, con el fin de conocer la situación actual de la organización.

Las personas a las cuales fue aplicada esta encuesta son: Propietaria, Bodeguero, Vendedor.

El modelo de encuesta aplicado es el siguiente:

FERRETERIA BOLIVARIANA

ENCUESTA

La presente encuesta servirá para recopilar información acerca del manejo de la información en la Ferreteria. Sírvase responder marcando con una X en la alternativa que considere responda la realidad del negocio.

CARGO: _____

1. ¿Qué cantidad de productos se maneja en bodega mensualmente?

    1. 1000-2000 ☐
    2. 2000-3000 ☐
    3. Más de 3000 ☐

2. ¿La empresa se proyecta en el futuro a la apertura de sucursales?

    Si ☐ No ☐

3. ¿Tiene su empresa clasificados los productos de alguna manera?

    Si ☐ No ☐

4. ¿Con cuántos proveedores cuenta la empresa?

    1. 0-10 ☐
    2. 10-20 ☐
    3. Más de 20 ☐

5. ¿Qué número de clientes atiende en promedio la ferretería diariamente?

    1. 0-20 ☐
    2. 20-40 ☐
    3. Más de 40 ☐

6. Seleccione los servicios que presta la empresa

    1. Venta de Productos de ferretería ☐
    2. Instalaciones Varias ☐

7. Determine en que porcentaje se registran pérdidas de productos (al mes)

    1. 0% ☐

2. <5% ☐
3. >5% ☐

8. Seleccione las razones de las pérdidas

1. Robo ☐
2. Falta de Control en Bodega ☐
3. Deterioro ☐

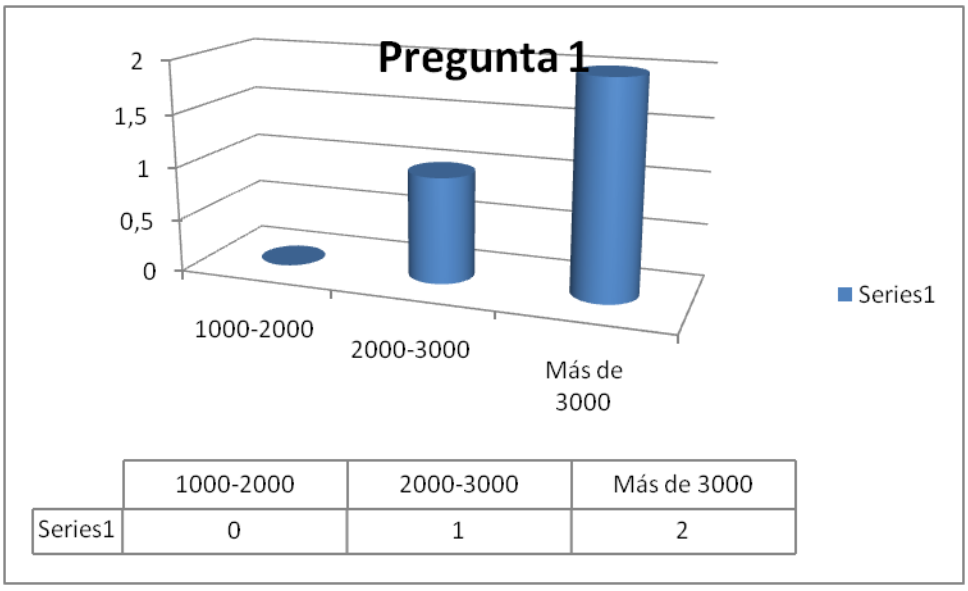
9. ¿Cómo se maneja la información de la empresa actualmente?

1. Manualmente ☐
2. Utilizando Hojas Electrónicas ☐
3. A través de un sistema informático ☐

4.2 Análisis de la Información

Para realizar la tabulación de la encuesta se utilizó Microsoft Excel, haciendo un análisis mediante las preguntas y respuestas más precisas y concretas.

1. ¿Qué cantidad de productos se maneja en bodega mensualmente?



	1000-2000	2000-3000	Más de 3000
Series1	0	1	2

Figura1. Pregunta1 Encuesta 1

Comentario: En vista de que el número de productos que maneja la ferretería mensualmente es más de 3000, no es eficiente un control manual de productos debido al margen de error que puede tener este control.

2. ¿La empresa tiene o tendrá sucursales?

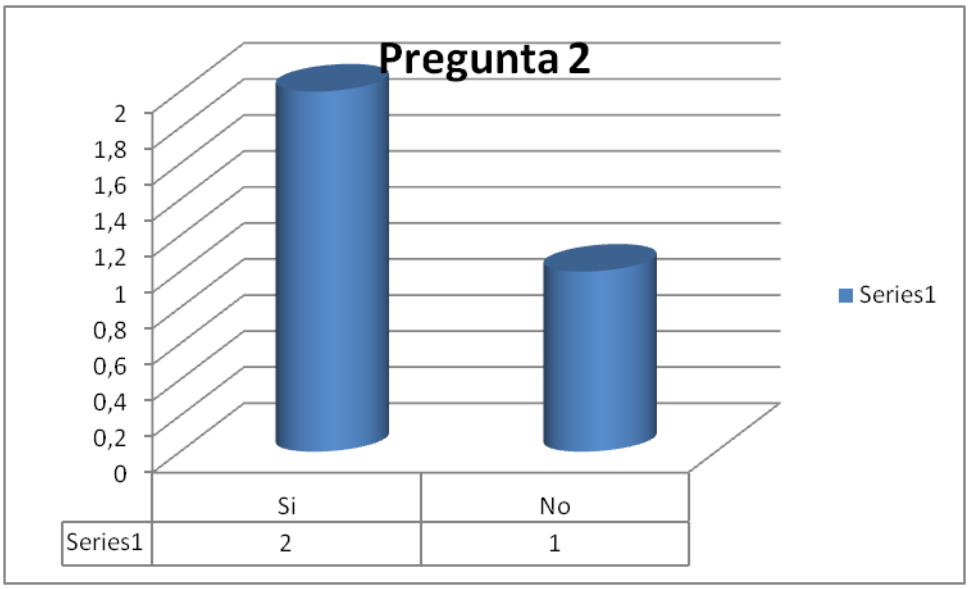


Figura2. Pregunta2 Encuesta 1

Comentario: La empresa si contara con sucursales en el futuro.

3. ¿Tiene su empresa clasificados los productos de alguna manera?

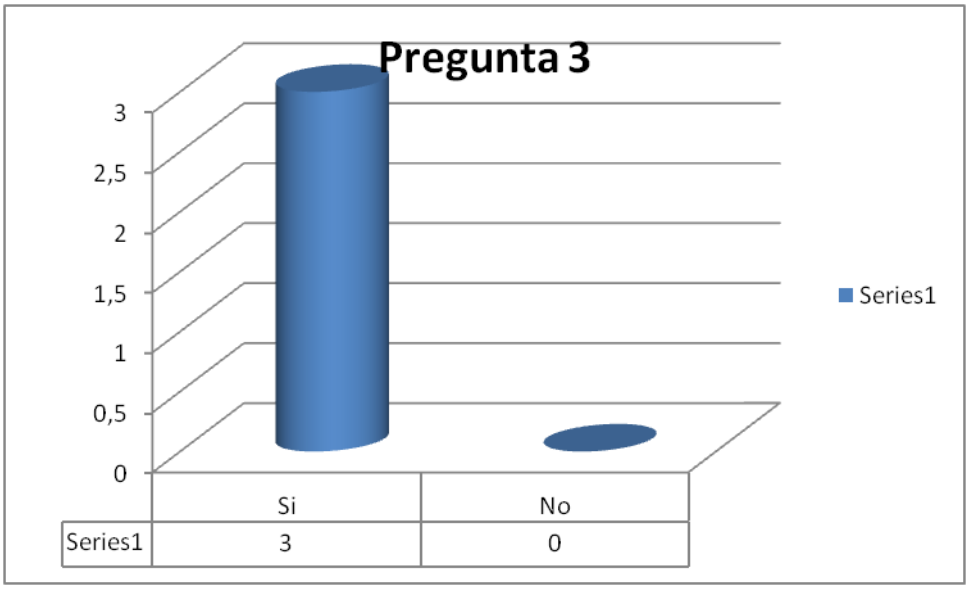


Figura3. Pregunta3 Encuesta 1

Comentario: En la Ferretería Bolivariana se tiene clasificados los productos por categorías como por ejemplo: Material Eléctrico, material de construcción, pinturas, pernos, tornillos, mangueras, etc.

4. ¿Con cuántos proveedores cuenta la empresa?

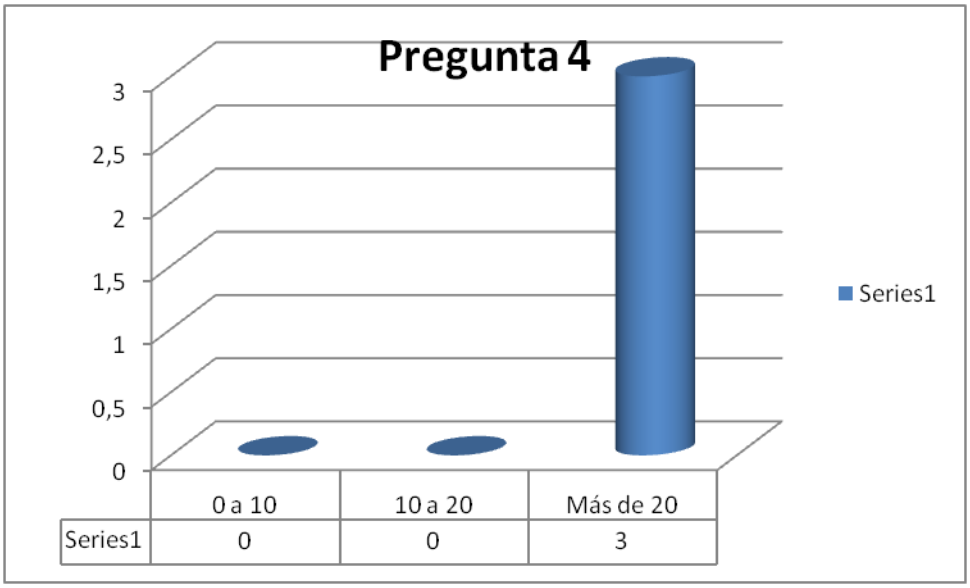


Figura4. Pregunta4 Encuesta 1

Comentario: La ferretería cuanta con más de 20 proveedores

5. ¿Qué número de clientes atiende en promedio la ferretería diariamente?

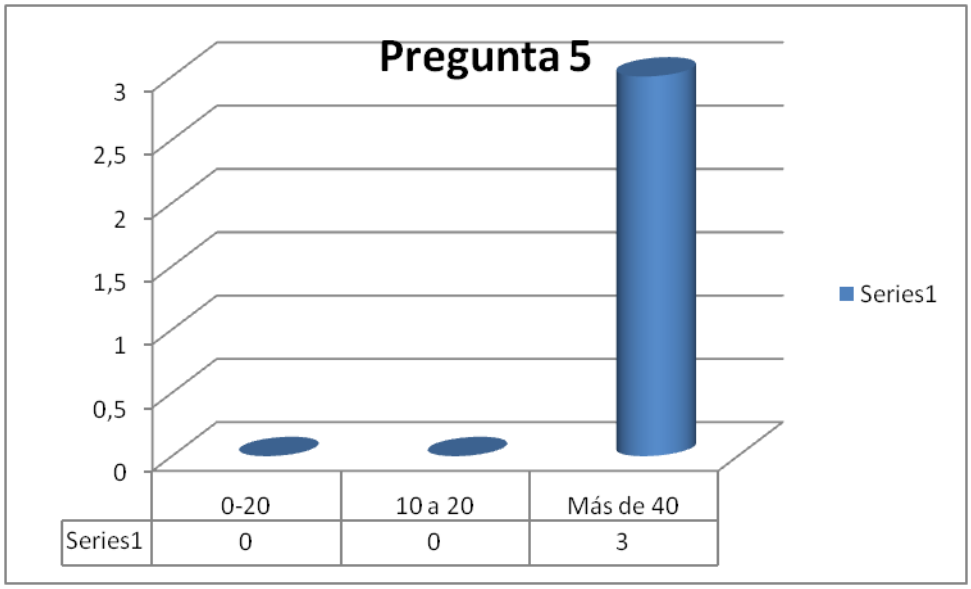


Figura5. Pregunta5 Encuesta 1

Comentario: La Ferretería Bolivariana atiende un promedio de más de 40 clientes diarios.

6. Seleccione los servicios que presta la empresa

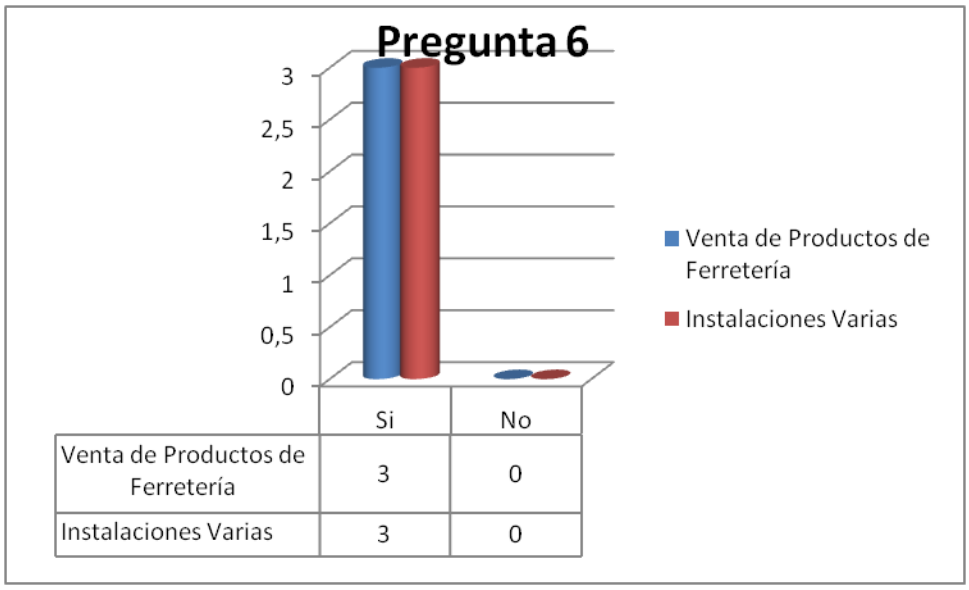


Figura6. Pregunta6 Encuesta 1

Comentario: La Ferretería Bolivariana vende productos de ferretería en general y también brinda servicios como instalaciones de calefones, electricidad, plomería, etc.

7. Determine en que porcentaje se registran pérdidas de productos (al mes)

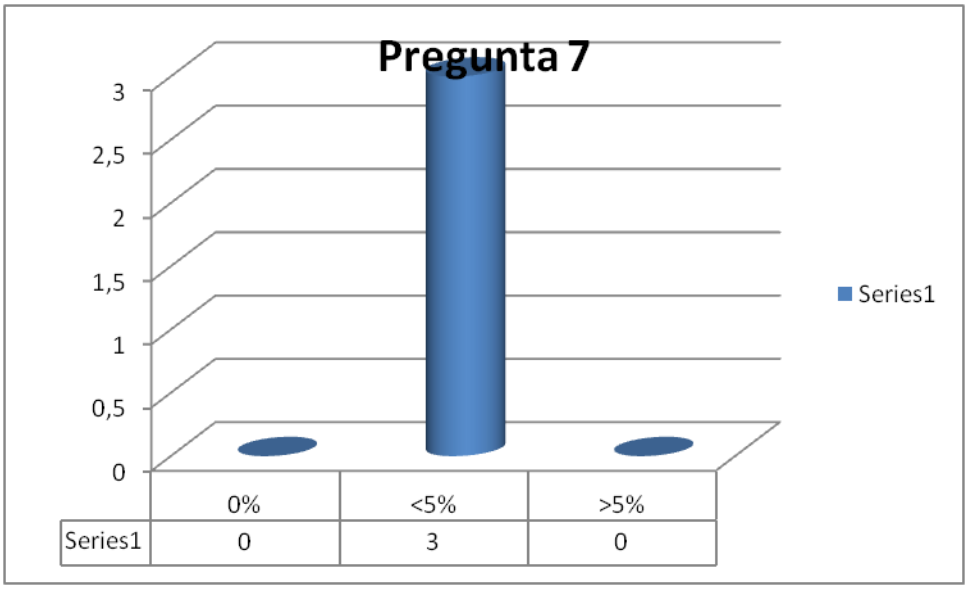


Figura7. Pregunta7 Encuesta 1

Comentario: Hay un porcentaje de pérdida de productos menos de 5% al mes.

8. Seleccione las razones de las pérdidas

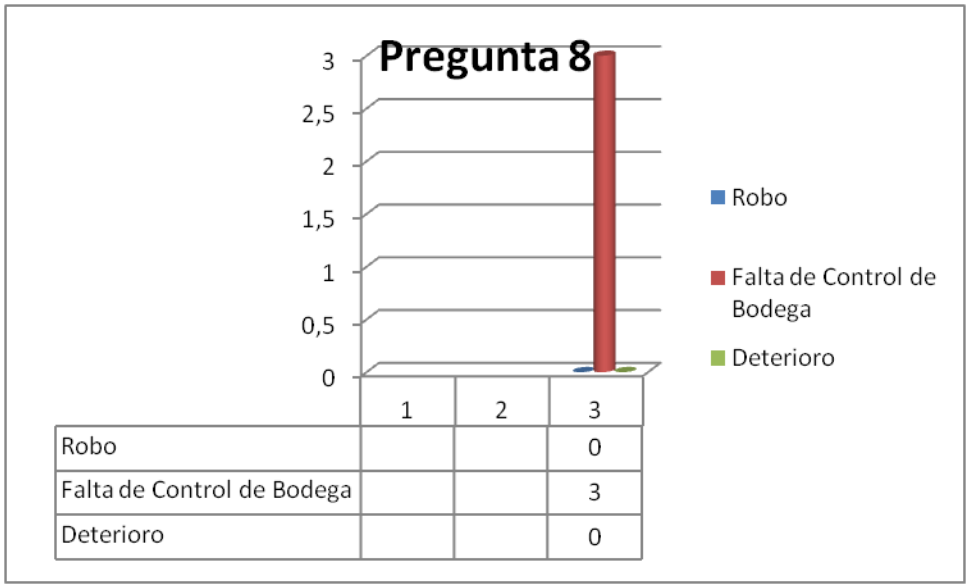
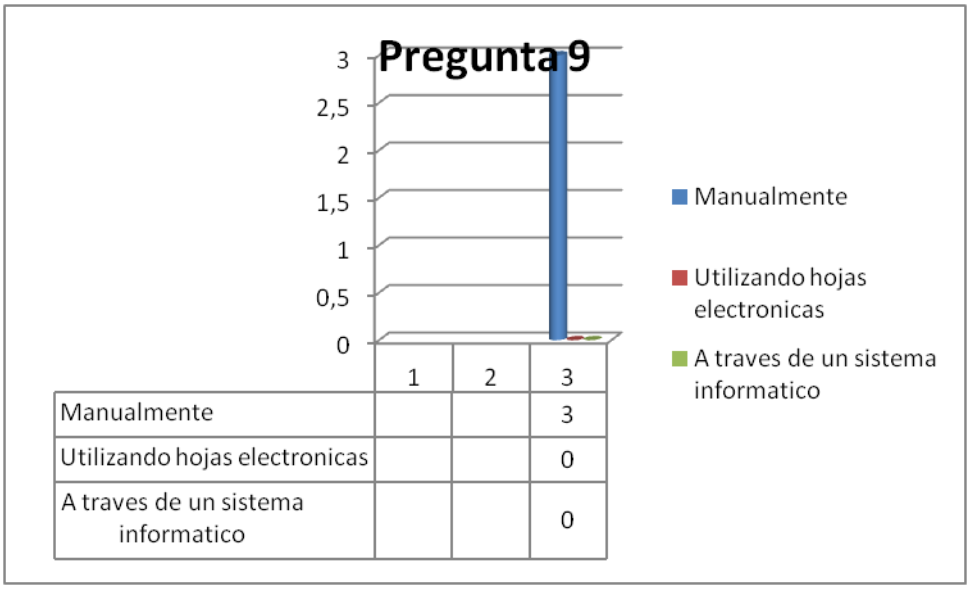


Figura8. Pregunta8 Encuesta 1

Comentario: Las pérdidas económicas de la empresa se deben al inadecuado control de existencias de productos en bodega ya que no se cuenta con un sistema que permita mantenernos informados constantemente de la existencia de cada producto que tiene la empresa en stock.

9. ¿Cómo se maneja la información de la empresa actualmente?



	1	2	3
Manualmente			3
Utilizando hojas electronicas			0
A traves de un sistema informatico			0

Figura9. Pregunta9 Encuesta 1

Comentario: La empresa maneja la información manualmente, en vista de que la empresa cuenta con más de 24000 productos se debe llevar la información de una manera más ordenada y segura.

4.3 Interpretación de Resultados

Dado los resultados obtenidos mediante el método de inferencia lógica Ponendo Ponens (Que afirmando afirma) está justificado el desarrollo del sistema: "Sistema de Control de Productos y Servicios de la Ferretería Bolivariana para evitar Pérdidas Económicas ", ya que este si evitara pérdidas económicas en la Ferreteria Bolivariana.

CAPITULO V

5.1 CONCLUSIONES

Al finalizar este proyecto podemos concluir lo siguiente:

- A través del uso de un Sistema Automatizado para facturación y control de existencias se cumplirá con las expectativas de los socios, ya que la atención será más efectiva.
- En el caso de que la población aumentara, el sistema está apto para cubrir con las expectativas de todos los usuario nuevos, es decir este programa es apto de expansibilidad.
- La implementación de un sistema automático mejora notablemente el manejo de la información en forma manual, aun cuando tome un poco de tiempo antes de adaptarse a la nueva forma de trabajo.

5.2 RECOMENDACIONES:

- Se debe hacer un respaldo de la Información una vez por mes para evitar perdida de información ya que los equipos están expuestos a daños físicos y lógicos.
- Todas las personas deben estar capacitadas sobre el funcionamiento del "Sistema de Control de Productos y Servicios de la Ferretería Bolivariana para evitar Pérdidas Económicas", para poder optimizar el tiempo.
- Se debe tener presente que el sistema debe trabajar ininterrumpidamente, es decir se debe procurar que el sistema funcione al 100%, hay que recordar que todo puede fallar menos el sistema, esto evitara inconvenientes al usuario como al programador.

CAPITULO VI

PROPUESTA

6.1 Datos Informativos

Instituciones:

- Universidad Técnica de Ambato, Facultad de Ingeniería en Sistemas, Electrónica e Industrial, Carrera de Ingeniería en Sistemas Computacionales e Informáticos
- Ferretería Bolivariana

Coordinadora Empresarial: Janeth Gavilanes

Tutora: Ing. Teresa Freire

Investigadora: Srta. Alexandra Patricia Solís Gavilanes

6.1 Antecedentes de la Propuesta

La Ferretería Bolivariana realiza todas sus funciones como facturación de venta, retenciones de compra, consultas de precios de productos con procesos manuales, lo que genera confusión de la información y pérdida de documentos físicos. Además, la búsqueda de los documentos como facturas de compra para consultar los precios de los productos provoca que la atención a los clientes se la realice con demora. Por estas razones surge la

necesidad de desarrollar e implementar un sistema de control de productos y servicios para poder optimizar el tiempo y satisfacer las necesidades de los clientes.

El sistema está realizado en base a las leyes y reglamentos del SRI para una empresa que lleva contabilidad de forma mensual.

6.3 Justificación

La implantación de un sistema de control de productos y servicios tiene como objetivo organizar, manipular, almacenar de manera ordenada y optimizar el tiempo en la realización de la factura de venta, retención de compra, consultas de clientes, proveedores, productos, precios de los productos, llevar un control de las existencias de los productos y mantener a la propietaria informada del control de sus cuentas mediante reportes, a través de la sistematización de los procesos para satisfacer las necesidades tanto de la propietaria como de los clientes de la Ferretería Bolivariana y así poder brindar servicio de calidad.

6.4 Objetivos

6.4.1 Objetivo General

Diseñar y desarrollar un Sistema de Control de Productos y Servicios para Evitar Pérdidas Económicas en la Ferretería Bolivariana.

6.4.2 Objetivos Específicos

- Diseñar los formularios para el ingreso, consulta y búsquedas de los clientes, proveedores, productos, cheques, factura de compra, factura de venta, retención en compra, retención en ventas.
- Realizar pruebas para verificar el funcionamiento del sistema.

- Implantar el Sistema de Control de Productos y Servicios para Evitar Pérdidas Económicas en la Ferretería Bolivariana.
- Brindar capacitación a los usuarios para el manejo del sistema de control de productos y servicios.

6.5 Fundamentación

6.5.1 Programación orientada a objetos

La **Programación Orientada a Objetos** (**POO** u **OOP** según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

Los objetos son entidades que combinan *estado, comportamiento e identidad*:

- El *estado* está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El *comportamiento* está definido por los procedimientos o *métodos* con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La *identidad* es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener, reutilizar y volver a utilizar.

De aquella forma, un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los

objetos disponen de mecanismos de interacción llamados métodos que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan ni deben separarse el estado y el comportamiento.

Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a ninguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una programación estructurada camuflada en un lenguaje de programación orientado a objetos.

Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean éste nuevo paradigma, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

6.5.1.1 Características de la POO

Hay un cierto acuerdo sobre exactamente qué características de un método de programación o lenguaje le definen como "orientado a objetos", pero hay un consenso general en que las características siguientes son las más importantes:

Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos

en el sistema sin revelar *cómo* se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.

Encapsulamiento: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

Principio de ocultación: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una *interfaz* a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

Polimorfismo: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama *asignación tardía* o *asignación dinámica*. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

Herencia: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos

especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en *clases* y estas en *árboles* o *enrejados* que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay *herencia múltiple*.

Recolección de basura: la Recolección de basura o Garbage Collection es la técnica por la cual el ambiente de Objetos se encarga de destruir automáticamente, y por tanto desasignar de la memoria, los Objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo Objeto y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse manualmente. [4]

6.5.2 Microsoft Visual Studio

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

A partir de la versión 2005 Microsoft ofrece gratuitamente las *Express Editions*. Estas son varias ediciones básicas separadas por lenguajes de programación o plataforma enfocadas

[4] http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

para novatos y entusiastas. Estas ediciones son iguales al entorno de desarrollo comercial pero sin características avanzadas. Las ediciones que hay son:

- Visual Basic Express Edition
- Visual C# Express Edition
- Visual C++ Express Edition
- Visual J# Express Edition (Desapareció en Visual Studio 2008)
- Visual Web Developer Express Edition (para programar en ASP.NET)

Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de MS SQL Server llamada SQL Server Express Edition cuyas principales limitaciones son que no soporta bases de datos superiores a 4 GB de tamaño, únicamente utiliza un procesador y un Gb de Ram, y no cuenta con el Agente de SQL Server.

Visual Studio 2005 se empezó a comercializar a través de Internet a partir del 4 de Octubre de 2005 y llegó a los comercios a finales del mes de Octubre en inglés. En castellano no salió hasta el 4 de Febrero de 2006. Microsoft eliminó *.NET*, pero eso no indica que se alejara de la plataforma .NET, de la cual se incluyó la versión 2.0.

La actualización más importante que recibieron los lenguajes de programación fue la inclusión de *tipos genéricos*, similares en muchos aspectos a las plantillas de C#. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

Se incluye un diseñador de implantación, que permite que el diseño de la aplicación sea validado antes de su implantación. También se incluye un entorno para publicación web y pruebas de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.

Visual Studio 2005 también añade soporte de 64-bit. Aunque el entorno de desarrollo sigue siendo una aplicación de 32 bits Visual C++ 2005 soporta compilación para x86-64

(AMD64 e Intel 64) e IA-64 (Itanium). El SDK incluye compiladores de 64 bits así como versiones de 64 bits de las librerías.

Visual Studio 2005 tiene varias ediciones radicalmente distintas entre sí: Express, Standard, Professional, Tools for Office, y 5 ediciones Visual Studio Team System. Éstas últimas se proporcionaban conjuntamente con suscripciones a MSDN cubriendo los 4 principales roles de la programación: Architects, Software Developers, Testers, y Database Professionals. La funcionalidad combinada de las 4 ediciones Team System se ofrecía como la edición Team Suite.

Tools for the Microsoft Office System está diseñada para extender la funcionalidad a Microsoft Office.

Las ediciones Express se han diseñado para principiantes, aficionados y pequeños negocios, todas disponibles gratuitamente a través de la página de Microsoft se incluye una edición independiente para cada lenguaje: Visual Basic, Visual C++, Visual C#, Visual J# para programación .NET en Windows, y Visual Web Developer para la creación de sitios web ASP.NET. Las ediciones express carecen de algunas herramientas avanzadas de programación así como de opciones de extensibilidad.

Se lanzó el service Pack 1 para Visual Studio 2005 el 14 de Diciembre de 2006.

La versión interna de Visual Studio 2005 es la 8.0, mientras que el formato del archivo es el 9.0.[5]

6.5.3 .Net Framework

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa

[5] http://msdn.microsoft.com/es-es/library/ms225360(VS.80).aspx

intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

6.5.3.1 Componentes de .Net Framework

Los principales componentes del marco de trabajo son:

- El conjunto de lenguajes de programación
- La Biblioteca de Clases Base o BCL
- El Entorno Común de Ejecución para Lenguajes o CLR por sus siglas en inglés.

6.5.3.1.1 Lenguajes de programación

Debido a la publicación de la norma para la **infraestructura común de lenguajes** (*CLI* por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Algunos de los lenguajes desarrollados para el **marco de trabajo .NET** son: C#, Visual Basic, Delphi (Object Pascal), C++, J#, Perl, Python, Fortran, Cobol y PowerBuilder.

6.5.3.1.2 Biblioteca de clases base

La Biblioteca de Clases Base (BCL por sus siglas en inglés) maneja la mayoría de las operaciones básicas que se encuentran involucradas en el desarrollo de aplicaciones, incluyendo entre otras:

- Interacción con los dispositivos periféricos
- Manejo de datos (ADO.NET)
- Administración de memoria
- Cifrado de datos
- Transmisión y recepción de datos por distintos medios (XML, TCP/IP)
- Administración de componentes Web que corren tanto en el servidor como en el cliente (ASP.NET)
- Manejo y administración de excepciones
- Manejo del sistema de ventanas
- Herramientas de despliegue de gráficos (GDI+)
- Herramientas de seguridad e integración con la seguridad del sistema operativo
- Manejo de tipos de datos unificado
- Interacción con otras aplicaciones
- Manejo de cadenas de caracteres y expresiones regulares
- Operaciones aritméticas
- Manipulación de fechas, zonas horarias y periodos de tiempo
- Manejo de arreglos de datos y colecciones
- Manipulación de archivos de imágenes
- Aleatoriedad
- Generación de código
- Manejo de idiomas
- Auto descripción de código
- Interacción con el API Win32 o Windows API.
- Compilación de código

Esta funcionalidad se encuentra organizada por medio de espacios de nombres jerárquicos.

La Biblioteca de Clases Base se clasifica, en cuatro grupos clave:

- ASP.NET y Servicios Web XML
- Windows Forms
- ADO.NET
- .NET

6.5.3.1.3 Common Language Runtime (CLR)

Estructura interna del entorno de ejecución en lenguaje común *(CLR por sus siglas en inglés).*

Figura10. Modelo CLR 1

Este es el lenguaje insignia de **.NET Framework** *(marco de trabajo .NET)* y pretende reunir las ventajas de lenguajes como C, C++ y Visual Basic en uno solo. El CLR es el verdadero núcleo del framework de .NET, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo (W2k y W2003).

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .NET en un código intermedio, el MSIL *(Microsoft Intermediate Lenguaje)*, similar al BYTECODE de Java. Para generarlo, el compilador se basa en la especificación CLS *(Common Language Specification)* que determina las reglas necesarias para crear el código MSIL compatible con el CLR.

Para ejecutarse se necesita un segundo paso, un compilador JIT *(Just-In-Time)* es el que genera el código máquina real que se ejecuta en la plataforma del cliente. De esta forma se

consigue con .NET independencia de la plataforma de hardware. La compilación JIT la realiza el CLR a medida que el programa invoca métodos. El código ejecutable obtenido se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.

6.5.3.2 Características

Es el encargado de proveer lo que se llama código administrado, es decir, un entorno que provee servicios automáticos al código que se ejecuta. Los servicios son variados:

- Cargador de clases: permite cargar en memoria las clases.
- Compilador MSIL a nativo: transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta.
- Administrador de código: coordina toda la operación de los distintos subsistemas del Common Language Runtime.
- Recolector de basura: elimina de memoria objetos no utilizados.
- Motor de seguridad: administra la seguridad del código que se ejecuta.
- Motor de depuración: permite hacer un seguimiento de la ejecución del código aun cuando se utilicen lenguajes distintos.
- Verificador de tipos: controla que las variables de la aplicación usen el área de memoria que tienen asignado.
- Administrador de excepciones: maneja los errores que se producen durante la ejecución del código.
- Soporte de multiproceso (hilos): permite ejecutar código en forma paralela.
- Empaquetador de COM: coordina la comunicación con los componentes COM para que puedan ser usados por el .NET Framework.

- Soporte de la Biblioteca de Clases Base: interfaz con las clases base del .NET Framework. Esto quiere decir que existen tipos de estructuras como es la de java y la .NET [6]

6.5.4 Crystal Reports

Es una aplicación de inteligencia empresarial utilizada para diseñar y generar informes desde una amplia gamas de fuentes de datos (bases de datos).

Varias otras aplicaciones, como Microsoft Visual Studio, incluyen una versión OEM de *Crystal Reports* como una herramienta de propósito general del informes/reportes. Crystal Reports se convirtió en el escritor de informes estándar cuando Microsoft lo liberó con Visual Basic.

Los usuarios al instalar Crystal Reports en un equipo y utilizarlo para seleccionar *filas y columnas* específicas de una tabla de datos compatibles, pueden organizar los datos en el informe en el formato que necesiten. Una vez que el diseño está completo, el informe se puede guardar/salvar como un archivo con extensión rpt. Se puede acceder nuevamente al informe reabriendo el mismo, y poder *refrescar* los datos. Si la fuente de base de datos se ha actualizado, el informe se *refrescará* reflejando estas actualizaciones.

6.5.4.1 Secciones de diseño

Crystal Reports posee 5 secciones dentro de la plantilla de diseño:

Sección	**Características**
Encabezado del informe (EI)	Sección usada para el *título del informe*, o bien, cualquier otra información que el usuario desee que aparezca en esa parte. Por otro

[6] http://es.wikipedia.org/wiki/Microsoft_.NET

	lado, puede ser usada para insertar gráficos y tablas cruzadas, los cuales incluyen datos para todo el informe.
Encabezado de página (EP)	Sección usada para agregar la información que se desea que aparezca en la *parte superior* de cada página. Pueden ser *nombres de capítulos*, *nombre del documento*, etc. Se puedes desplegar *títulos de campo* sobre los campos (columnas) mismos en su informe.
Detalles (D)	Sección usada para el *cuerpo del informe* y se imprime una vez por registro (fila). La mayor parte de los datos del informe aparece en esta sección.
Pie de informe (PI)	Sección usada para la información que se desea que aparezca *sólo una vez* al *final del informe* (por ejemplo, *totales generales*) y para los *gráficos* y las *tablas cruzadas* que incluyen datos relativos a todo el informe.
Pie de página (PP)	Sección que contiene el *número de página* y cualquier otra información que se desea que aparezca en la parte inferior de la misma.

Tabla 3.Secciones de Diseño de Crystal Report

6.5.4.2 Grupos

Si se añade un *grupo, resumen* o *subtotal* al informe, el programa crea dos secciones más:

Sección	**Características**
Encabezado de grupo	Sección que contiene el *campo de nombre de grupo*. Se usar para mostrar *gráficos* o *tablas cruzadas* de datos específicos del grupo. *Sólo se imprime una vez al principio de un grupo*.
Pie de grupo	Esta sección incluye el *valor de resumen*, si lo hay, y se puede usar para *insertar gráficos* y *tablas cruzadas*. *Se imprime sólo una vez al final de un*

	grupo.

Tabla 4. Grupos en Crystal Report

Los grupos son unas especies de *filtros*, que permiten ordenar los datos según el criterio elegido, es decir, según el grupo específico elegido.

6.5.4.3 Informe en blanco

Una de las opciones para la creación de informes es comenzar desde cero. Para *cargar* las bases de datos existe un *Asistente de base de datos*, donde muestra los siguientes orígenes de datos soportados:

Origen de datos	**Características**
Conexiones actuales	Carpeta que muestra una lista de los orígenes de datos a los que está conectado actualmente.
Favoritos	Muestra una lista de los orígenes de datos que se utilizan normalmente y que se ha mantenido en la lista Favoritos.
Historial	Muestra una lista de los orígenes de datos utilizados recientemente. *Se muestran las últimas cinco fuentes de datos*.
Crear una nueva conexión	Muestra las subcarpetas de varios orígenes de datos a los que se puede conectar.
Access-Excel (DAO)	Access - dBASE 5.0, III y IV - Excel 3.0, 4.0 y 5.0 - Importación de HTML - Lotus WK1, WK3, WK4 - Paradox 3.x, 4.x, y 5.x - Texto
ADO.NET (XML)	
Archivos de base de	Excel: .xls, .xlw - Archivos XML: .xml, .xsd - Definición de campo:

datos:	.ttx - Archivos xBase: .dbf - Diccionarios Btrieve: .ddf - Archivos Access: .mdb - Archivos de vínculos de datos: .udl
Exchange 5.5 Message Tracking Log	
Exhange Message Tracking Log	
Legacy Exchange	
Mailbox Admin	
ODBC (RDO):	Base de datos Xtreme 11.5 - dBASE Files - Excel Files - MS Access Database - Xtreme Sample Database 11.5
Olap	
OLE DB (ADO)	Microsoft Jet 4.0 OLE DB - Microsoft Office 12.0 Access Database Engine OLE DB - Microsoft OLE DB for: Analysis Services 9.0 , Data Mining Services, Indexing Service, Internet Publishing, ODBC Drivers, OLAP Servides 8.0, Oracle, SQL Server, Simple - MSdataShape - Proveedor de bases de datos OLE para servicios de directorio de Microsoft
Outlook/Exchange	
Public Folder ACL	
Puclic Folder Admin	

Public Folder Replica	
Registro de eventos actuales de NT	
Registro de eventos archivados NT	
Sólo definiciones de campo	
Universos	Se conecta a BusinessObjects Enterprise)
Web/IIS Log Files	
xBase (.dbf)	
XML	
Más orígenes de datos...	
Repositorio	Se conecta a BusinessObjects Enterprise). Muestra el contenido del *repositorio* mediante el Explorador de BusinessObjects Enterprise. Al hacer clic en *Establecer nueva conexión*, para abrir el Explorador de BusinessObjects Enterprise; se puede seleccionar un comando SQL o una vista empresarial existentes.

Tabla 5. Características de Informe en Blanco de Crystal Report

6.5.4.4 Tablas múltiples

Si se ha creado un informe que posee datos de dos o más tablas, desde una base de datos en los formatos soportados, tendrá que vincular durante el proceso de elaboración del informe (antes de agregar el contenido al informe).

Agregar y vincular tablas múltiples

- Elegir el comando *Asistente de base de datos*, del menú *Base de datos* (aparece el cuadro de diálogo *Asistente de base de datos*).
- En la ficha Datos, se deben seleccionar las tablas que se desee agregar al informe (aparece la ficha *Vínculos* en el *Asistente de base de datos*.
- Hacer clic en la ficha *Vínculos* para que se muestren las bases de datos disponibles actualmente para establecer vínculos.
- Para crear los vínculos manualmente, de debe arrastrar un *campo* de una tabla hasta un *campo* de otra tabla. Si no se tiene éxito en la creación del vínculo, se recibe un mensaje.
- *Aceptar*, para finalizar el asistente.

6.5.4.5 Campos

Muchos de los datos que se pueden insertar en el informe son campos de base de datos seleccionada. Éstos mostrarán los datos tal como están almacenados en la base de datos.

- ✓ Para insertar campos se debe seguir el siguiente procedimiento:
- ✓ Ir a la barra de herramientas estándar, clic sobre *Explorador de campos*.

Aparece el cuadro de diálogo *Explorador de campos*.

- ✓ Expandir la carpeta *Campos de base de datos*, para ver todas las tablas seleccionadas en las bases de datos.
- ✓ Expandir las tablas de forma individual para ver todos los campos que contienen.
- ✓ Hacer clic sobre el campo que se quiere insertar en el informe.

- ✓ Hacer clic sobre el botón *Examinar* para revisar los valores del campo seleccionado.
- ✓ Hacer clic en *Insertar* en informe para colocarlo en el informe, o bien, hacer clic y arrastrar hasta un lugar deseado, preferentemente en la sección *"Detalles"*.

6.5.4.6 Campos de fórmula

Si se quieren desplegar datos que son valores calculados, tendrá que crear un *campo de fórmula* y ponerlo en informe, similarmente a lo anterior (arrastrar hasta el lugar deseado, o insertarlo).

Se cuenta con un amplio espectro de fórmulas disponibles para la creación de éstas. Las cuales pueden ser programadas o creadas según el lenguaje crystal, integrado al programa, o bien, utilizando la sintaxis de Basic (Visual Basic).

Ejemplos

Campos: {cliente.Nombre del cliente}, {proveedor.Nombre del proveedor}

Texto: "Entre comillas", "separados por comas"

Operadores: + (sumar), / (dividir), -x (negativo)

Funciones (las funciones realizan cálculos tales como promedio, suma y conteo. Las funciones disponibles se listan con sus argumentos y se organizan según su uso).

Round (x), Trim (x)

Estructuras de control: "If" y "Select", ciclos "For"

Valores de campo de grupo (Por ejemplo, se pueden usar valores de *campo de grupo* para buscar el porcentaje del *total general* aportado por cada grupo).

Average (campo, Cpocond), Sum (campo, Cpocond, "condición")

Mezcla de muchos parámetros:

If ({cliente. NOMBRE DEL CLIENTE} [1 to 2] = "Ab") Then "TRUE"

Else "FALSE"

If ({cliente. NOMBRE DEL CLIENTE}) [1 to 2] = "Ab" and ToText({cliente. ID DEL CLIENTE}) [1] = "6"

or ({cliente. NOMBRE DEL CLIENTE}) [1 to 2] = "Ba" and

ToText({cliente. ID DEL CLIENTE}) [1] = "5" Then "elegido" Else "no elegido"

6.6 Metodología de desarrollo

Para el desarrollo de la aplicación se utilizo la metodología **Orientada a Objetos** es un paradigma que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basada en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento entre otras.

6.6.1 Análisis

6.6.1.1 Análisis de factibilidad

La Ferretería Bolivariana cuenta con un manejo de información manual e inadecuado para su empresa lo cual se demuestra mediante pérdidas económicas por falta de productos y mal manejo de información no contaba por esto es factible desarrollar un sistema de control de productos y servicios que permita a los funcionarios optimizar el tiempo para brindar un mejor servicio al usuario y llevar la información de una manera ordenada y las cuentas detalladas constantemente.

Factibilidad Técnica

Para la creación del Sistema Informático se cuenta con las herramientas necesarias como: Visual Studio 2005 y Sql como motor de base de datos.

Software

- Lenguaje de desarrollo Microsoft Visual Studio 2005
- Crystal Report 9 para la creación de reportes
- Microsoft SQL Server 2005 (MANAGER) como Motor de Bases de Datos.

Hardware

La Ferretería Bolivariana cuenta con una computadora de escritorio que será utilizada como servidor y una laptop que será el cliente, crearemos una conexión de red para poder llevar a cabo este proyecto.

Características del servidor

El servidor está ubicado en el área de Ventas ya que el área de sistemas no cuenta con su propio departamento.

- Sistema Operativo Windows XP
- Disco duro de 40 GB
- Memoria RAM 1GB
- Procesador Intel 3.2 GB

Características del Cliente

- Laptop Dell Inspiron 1420
- Disco Duro 320 GB
- Memoria 3GB
- Procesador Core 2 Duo
-

Factibilidad Económica

El presente proyecto de investigación es factible de realizarlo en el ámbito económico, ya que la empresa cuenta con todos los recursos necesarios para la ejecución del sistema informático.

Factibilidad Operativa

El Sistema Informático de Control de Productos de la Ferretería Bolivariana cuenta con una interfaz grafica agradable y de fácil manejo para los usuarios, los formularios contienen todas las funcionalidades de un sistema contable, como son: Ingreso, eliminación, actualización y búsqueda de proveedores, clientes, compras, factura de venta, retenciones, así como presenta consultas con búsqueda inteligente de sus productos, clientes y proveedores; contando con un control de datos como validación de número de cedula, fechas, controles mediante mensajes para no salir sin guardar la información, no borrar registros que tengan información relacionada, etc. Además esta modelado en la arquitectura cliente-servidor que servirá para actualizar la base de datos (servidor) desde cualquier pc (cliente).

Para que los usuarios del sistema, puedan aprovechar al máximo las capacidades del mismo, estos recibirán un manual de usuario el cual detalla los procesos del sistema informático de control de productos y servicios de la Ferretería Bolivariana.

Con todos estos antecedentes se puede concluir que el sistema es factible desde el punto de vista operativo.

6.6.1.2 Recopilación de Información Preliminar

La recopilación de la información partió de la documentación que se manejaba de forma manual en la empresa.

Factura de Compra

Para la elaboración del sistema se recibió una factura de compra de la empresa, la cual lleva los siguientes datos.

- En el encabezado tenemos:
- El nombre de la empresa proveedora
- La dirección de la empresa
- El teléfono de la empresa
- El RUC (Registro Único de Contribuyente) de la empresa proveedora
- El numero de Factura
- La autorización del SRI

Por otra parte del encabezado tenemos:

- Nombre del cliente
- Dirección del cliente
- Ciudad
- Teléfono
- Plazo._ Es el número de meses que da crédito la empresa proveedora.
- Fecha
- Vendedor

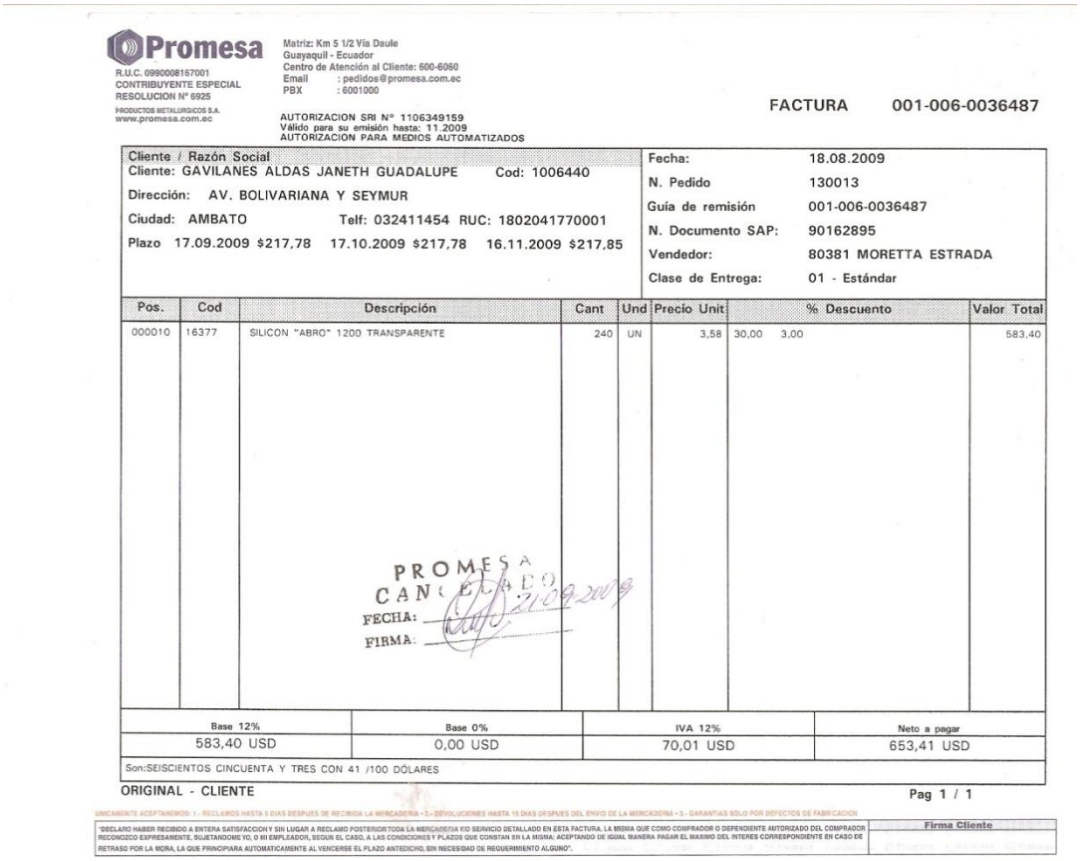
En el detalle tenemos:

- Pos
- Código
- Descripción._ Es el nombre del producto

- Cantidad._ Es el número de productos vendidos
- Unidad._ La unidad de medida del producto
- Precio Unitario._ Es el valor unitario del producto
- Descuento. El porcentaje de descuento
- Valor Total._ Es la cantidad multiplicada por el valor unitario del producto.

En la parte inferior tenemos:

- Subtotal._ Es la suma de los precios totales
- Iva._ Es el 12% del subtotal
- Total. _ Es la suma del Subtotal y el Iva



Promesa

R.U.C. 0990008167001
CONTRIBUYENTE ESPECIAL
RESOLUCION N° 5925
PRODUCTOS METALURGICOS S.A.
www.promesa.com.ec

Matriz: Km 5 1/2 Vía Daule
Guayaquil - Ecuador
Centro de Atención al Cliente: 600-6060
Email : pedidos@promesa.com.ec
PBX : 6001000

AUTORIZACION SRI N° 1106349159
Válido para su emisión hasta: 11.2009
AUTORIZACION PARA MEDIOS AUTOMATIZADOS

FACTURA 001-006-0036487

Cliente / Razón Social
Cliente: GAVILANES ALDAS JANETH GUADALUPE Cod: 1006440
Dirección: AV. BOLIVARIANA Y SEYMUR
Ciudad: AMBATO Telf: 032411454 RUC: 1802041770001
Plazo 17.09.2009 $217,78 17.10.2009 $217,78 16.11.2009 $217,85

Fecha: 18.08.2009
N. Pedido 130013
Guía de remisión 001-006-0036487
N. Documento SAP: 90162895
Vendedor: 80381 MORETTA ESTRADA
Clase de Entrega: 01 - Estándar

Pos.	Cod	Descripción	Cant	Und	Precio Unit	% Descuento	Valor Total
000010	16377	SILICON "ABRO" 1200 TRANSPARENTE	240	UN	3,58	30,00 3,00	583,40

PROMESA
CANCELADO
FECHA: 21/09/2009
FIRMA:

Base 12%	Base 0%	IVA 12%	Neto a pagar
583,40 USD	0,00 USD	70,01 USD	653,41 USD

Son:SEISCIENTOS CINCUENTA Y TRES CON 41 /100 DÓLARES

ORIGINAL - CLIENTE

Pag 1 / 1

Firma Cliente

Figura 11. Factura de Compra 1

Factura de Venta

Para la elaboración del sistema se recibió una factura de venta de la empresa, la cual lleva los siguientes datos.

En el encabezado tenemos:

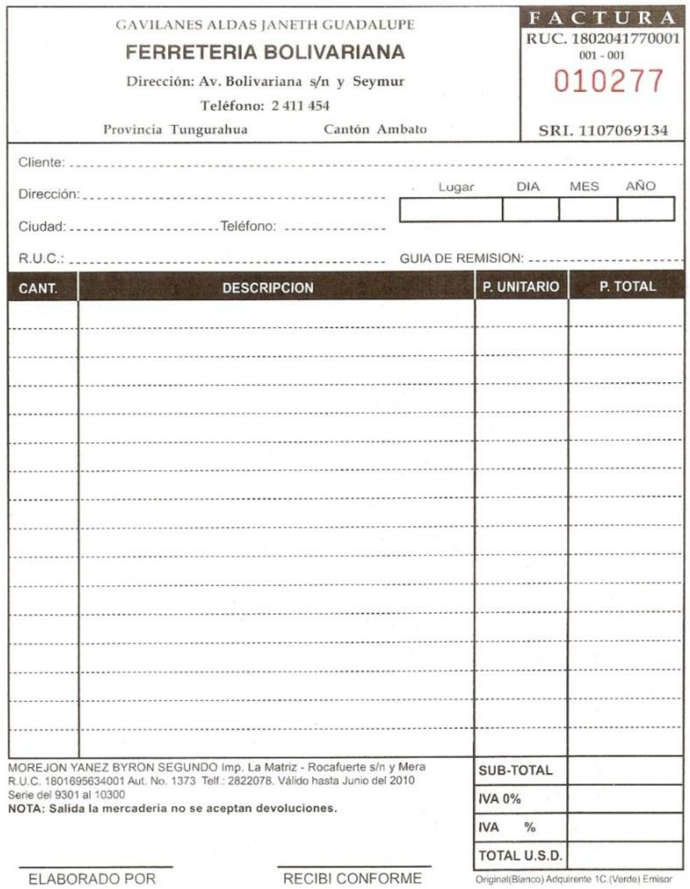
- El nombre de la propietaria
- El nombre de la empresa
- La dirección de la empresa
- El teléfono de la empresa
- La provincia y el Cantón de la ubicación de la empresa
- El RUC (Registro Único de Contribuyente) de la Ferretería
- El numero de Factura
- La autorización del SRI
- Lugar, Día, Mes, Año.- Es decir la fecha de elaboración de la factura.

En el detalle tenemos:

- Cantidad._ Es el número de productos vendidos
- Descripción._ Es el nombre del producto
- Precio Unitario._ Es el valor unitario del producto
- Precio Total._ Es la cantidad multiplicada por el valor unitario del producto.

En la parte inferior tenemos:

- Subtotal._ Es la suma de los precios totales
- Iva._ Es el 12% del subtotal
- Total. _ Es la suma del Subtotal y el Iva



GAVILANES ALDAS JANETH GUADALUPE

FERRETERIA BOLIVARIANA

Dirección: Av. Bolivariana s/n y Seymur

Teléfono: 2 411 454

Provincia Tungurahua Cantón Ambato

FACTURA

RUC. 1802041770001

001 - 001

010277

SRI. 1107069134

Cliente:

Dirección:

Lugar DIA MES AÑO

Ciudad: Teléfono:

R.U.C.: GUIA DE REMISION:

CANT.	DESCRIPCION	P. UNITARIO	P. TOTAL

MOREJON YANEZ BYRON SEGUNDO Imp. La Matriz - Rocafuerte s/n y Mera
R.U.C. 1801695634001 Aut. No. 1373 Telf.: 2822078. Válido hasta Junio del 2010
Serie del 9301 al 10300
NOTA: Salida la mercadería no se aceptan devoluciones.

SUB-TOTAL	
IVA 0%	
IVA %	
TOTAL U.S.D.	

ELABORADO POR RECIBI CONFORME

Original(Blanco) Adquirente 1C (Verde) Emisor

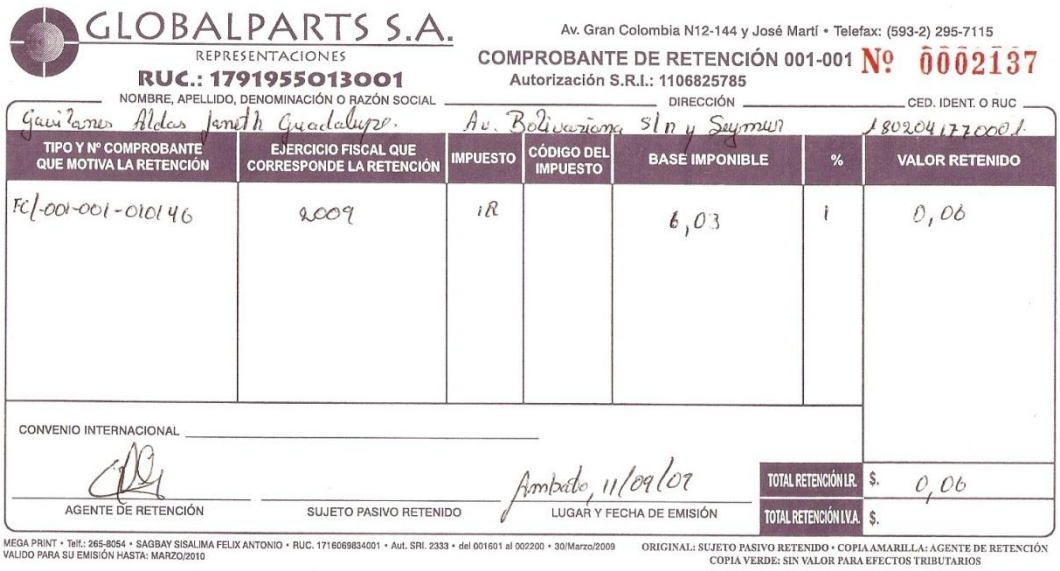
Figura 12. Factura de Venta 1

Retención en Venta

Este documento nos proporciona datos como:

- Ruc (Registro Único de Contribuyente) del cliente
- Nombre del cliente o de la empresa
- Dirección del cliente o de la empresa
- Número del Comprobante de Retención
- Tipo de Comprobante
- Fecha de Emisión
- Ejercicio Fiscal._ Es el año en que se realiza la retención

- Impuesto._ Es el nombre del impuesto a retener
- Código del impuesto
- Base Imponible._ Es el subtotal de la factura de venta
- %._ Es el porcentaje de descuento que corresponde al código del impuesto a retener.
- Valor retenido._ Es el % de la base imponible
- Total Retención._ Es la suma de los valores retenidos.



GLOBALPARTS S.A.
REPRESENTACIONES
RUC.: 1791955013001
Av. Gran Colombia N12-144 y José Martí • Telefax: (593-2) 295-7115
COMPROBANTE DE RETENCIÓN 001-001 Nº 0002137
Autorización S.R.I.: 1106825785

NOMBRE, APELLIDO, DENOMINACIÓN O RAZÓN SOCIAL: Gavilanes Aldas Janeth Guadalupe. | DIRECCIÓN: Av. Bolivariana s/n y Seymur | CED. IDENT. O RUC: 1802041770001

TIPO Y Nº COMPROBANTE QUE MOTIVA LA RETENCIÓN	EJERCICIO FISCAL QUE CORRESPONDE LA RETENCIÓN	IMPUESTO	CÓDIGO DEL IMPUESTO	BASE IMPONIBLE	%	VALOR RETENIDO
FC-001-001-010146	2009	IR		6,03	1	0,06

CONVENIO INTERNACIONAL

AGENTE DE RETENCIÓN | SUJETO PASIVO RETENIDO | LUGAR Y FECHA DE EMISIÓN: Ambato, 11/09/09

TOTAL RETENCIÓN I.R. $. 0,06
TOTAL RETENCIÓN I.V.A. $.

MEGA PRINT • Telf.: 265-8054 • SAGBAY SISALIMA FELIX ANTONIO • RUC. 1716069834001 • Aut. SRI. 2333 • del 001601 al 002200 • 30/Marzo/2009
VALIDO PARA SU EMISIÓN HASTA: MARZO/2010

ORIGINAL: SUJETO PASIVO RETENIDO • COPIA AMARILLA: AGENTE DE RETENCIÓN
COPIA VERDE: SIN VALOR PARA EFECTOS TRIBUTARIOS

Figura 13. Retención de Venta 1

Retención en Compra

También tenemos como documento la retención de compra, que es el que nosotros entregamos a nuestros proveedores

En el encabezado tenemos los siguientes datos:

- Nombre de la propietaria de la Ferretería Bolivariana
- El nombre de la Empresa "Ferretería Bolivariana"
- Dirección de la empresa

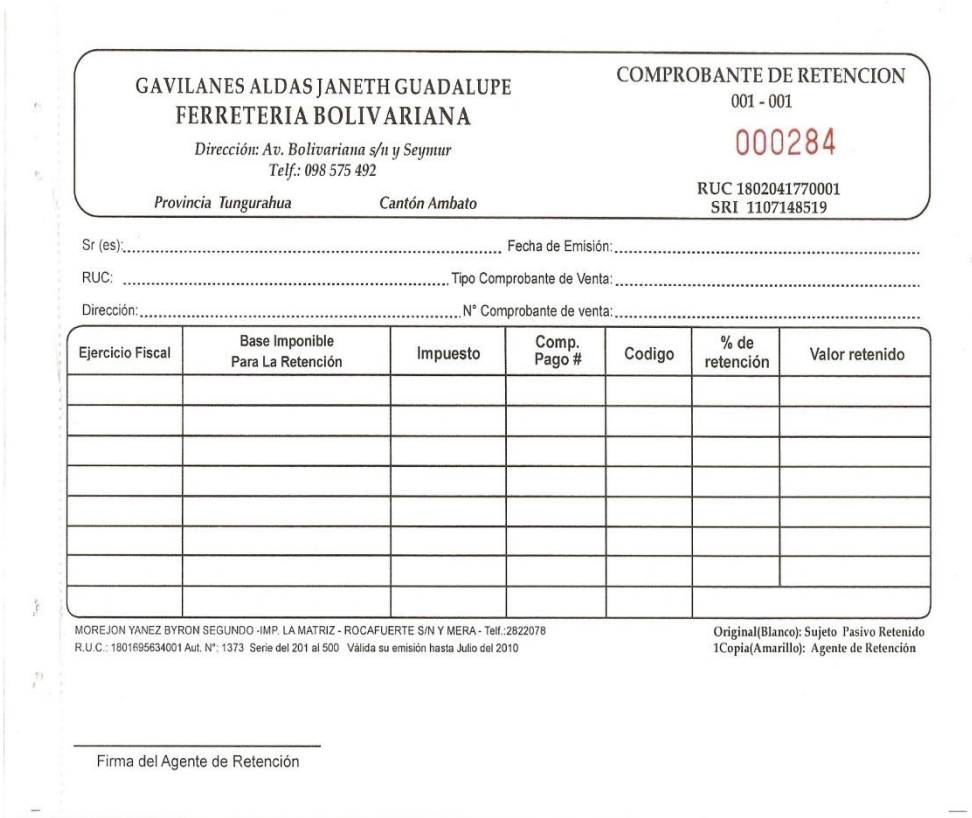
- Teléfono de la empresa
- La provincia y el cantón de la ubicación de la empresa
- El número de comprobante de retención
- El RUC (Registro Único de Contribuyente) de la empresa "Ferretería Bolivariana"
- La autorización del SRI

Por otra parte tenemos:

- Sr. (es)._ El nombre de nuestro proveedor o de la empresa que provee.
- RUC._ Es el RUC (Registro Único de Contribuyente) de nuestro proveedor
- Dirección del proveedor
- Número del Comprobante de Retención
- Tipo de Comprobante
- Fecha de Emisión

En el detalle tenemos:

- Ejercicio Fiscal._ Es el año en que se realiza la retención
- Base Imponible para la Retención._ Es el subtotal de la factura de venta
- Impuesto._ Es el nombre del impuesto a retener
- Comprobante de Pago
- Código del impuesto
- % de Retención._ Es el porcentaje de descuento que corresponde al código del impuesto a retener.
- Valor retenido._ Es el % de la base imponible
- Total Retención._ Es la suma de los valores retenidos.



GAVILANES ALDAS JANETH GUADALUPE
FERRETERIA BOLIVARIANA
Dirección: Av. Bolivariana s/n y Seymur
Telf.: 098 575 492
Provincia Tungurahua *Cantón Ambato*

COMPROBANTE DE RETENCION
001 - 001
000284
RUC 1802041770001
SRI 1107148519

Sr (es):..... Fecha de Emisión:.....
RUC: ..... Tipo Comprobante de Venta:.....
Dirección:..... N° Comprobante de venta:.....

Ejercicio Fiscal	Base Imponible Para La Retención	Impuesto	Comp. Pago #	Codigo	% de retención	Valor retenido

MOREJON YANEZ BYRON SEGUNDO -IMP. LA MATRIZ - ROCAFUERTE S/N Y MERA - Telf.:2822078
R.U.C.: 1801695634001 Aut. N°: 1373 Serie del 201 al 500 Válida su emisión hasta Julio del 2010

Original(Blanco): Sujeto Pasivo Retenido
1Copia(Amarillo): Agente de Retención

Firma del Agente de Retención

Figura 14. Retención de Compra 1

6.6.1.3 Análisis de Requisitos del Sistema

Debido a las necesidades de la propietaria el sistema requiere hacer Ingresos de Compras, Ventas, Categorías, Productos, Proveedores, Clientes, Cheques, Retención en Compras y Retención en Ventas; también se puede Consultar por medio de búsquedas inteligentes los registros de Categorías, Productos, Proveedores, Clientes, Compras, Ventas, Existencias, Cheques, Retención en Compras y Retención en Ventas; además genera Reportes de Productos, Proveedores, Clientes, Factura de Compra, Factura de Venta, Existencias en Cero, Compras Totales, Ventas Totales.

Se divide en los siguientes módulos:

Factura de Compra.- Este formulario permite agregar, editar, eliminar, guardar, cancelar y cerrar las facturas de compra que tiene la empresa, además actualiza las existencias e imprime la factura como un reporte.

Facturas de Venta.- Este formulario permite crear la factura de la empresa, donde se tiene funciones como Agregar, Editar, Eliminar, Guardar, Cancelar, Cerrar, actualizar las existencias de los productos, imprimir la factura como un reporte.

Categorías.- Este formulario permite Agregar, Editar, Eliminar, Guardar una Categoría o clasificaciones de productos que tenga la ferretería.

Clientes.- Este formulario permite ingresar, eliminar o modificar los datos de los clientes de la empresa, si en caso se ingresara un número de cédula no válido el sistema rechazará el ingreso.

Proveedores.- Este formulario nos sirve para ingresar o eliminar proveedores, así como también actualizar los datos informativos de nuestros proveedores.

Productos.- Este modulo permite ingresar los productos nuevos, también se puede buscar un producto dentro de este formulario, poder editarlo y actualizar los precios.

6.6.1.4 Diagramas UML

Los diagramas UML son un conjunto de herramientas, que nos permiten modelar, analizar y diseñar sistemas orientados a objetos, es decir la forma en la cual se relacionan el sistema con agentes externos como Usuarios.

6.6.1.4.1 Diagrama De Casos De Uso

Los Casos de Uso no son parte del diseño (cómo), sino parte del análisis (qué). De forma que al ser parte del análisis nos ayudan a describir qué es lo que el sistema debe

hacer. Los Casos de Uso son qué hace el sistema desde el punto de vista del usuario. Es decir, describen un uso del sistema y cómo este interactúa con el usuario. [7]

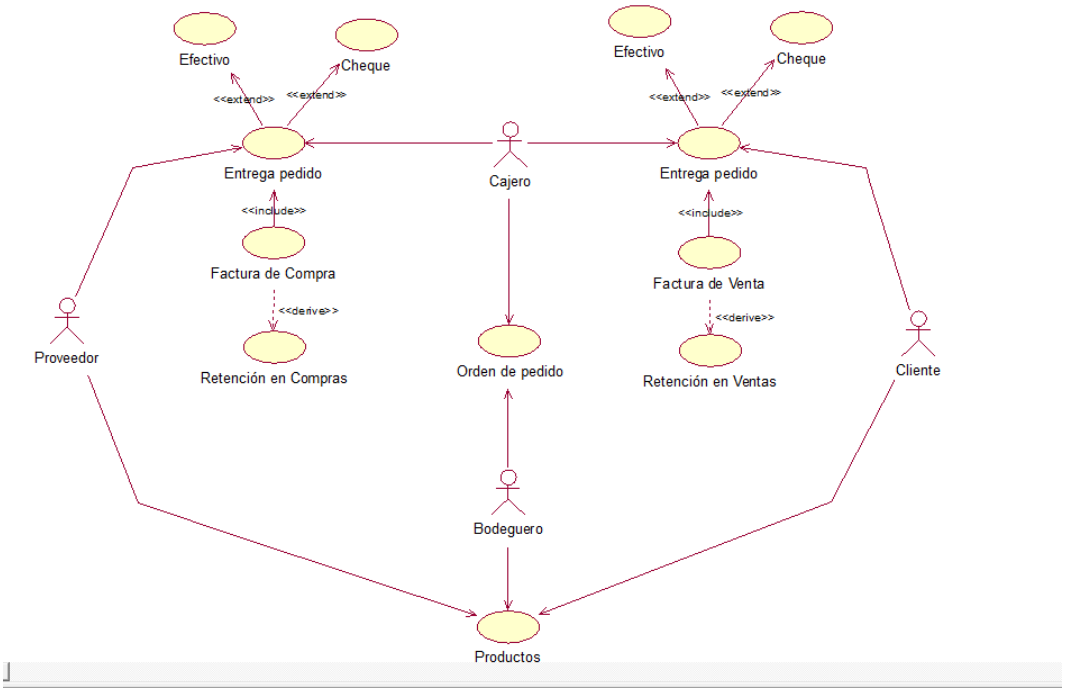


Figura15. Diagrama Casos de Uso 1

En nuestro diagrama de caso de usos la empresa hace el pedido al proveedor, posteriormente el proveedor entrega el pedido al bodeguero el cual se encarga de registrar los productos, la empresa cancela en efectivo o en cheque entregándole el proveedor a la empresa la factura de compra, por su parte la empresa entrega la retención de compra al proveedor y finalmente el proveedor entrega los productos a bodega.

Por otra parte el cliente hace el pedido al vendedor de la ferretería, el cliente paga en efectivo o cheque y el vendedor pasa al bodeguero la orden de pedido y entrega el producto al cliente con su respectiva factura de venta, el cliente puede o no entregar la retención de venta.

[7] http://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso

6.6.1.4.2 Diagrama de Clases

Este diagrama describe la estructura del sistema mostrando sus clases, atributos y las relaciones[8]

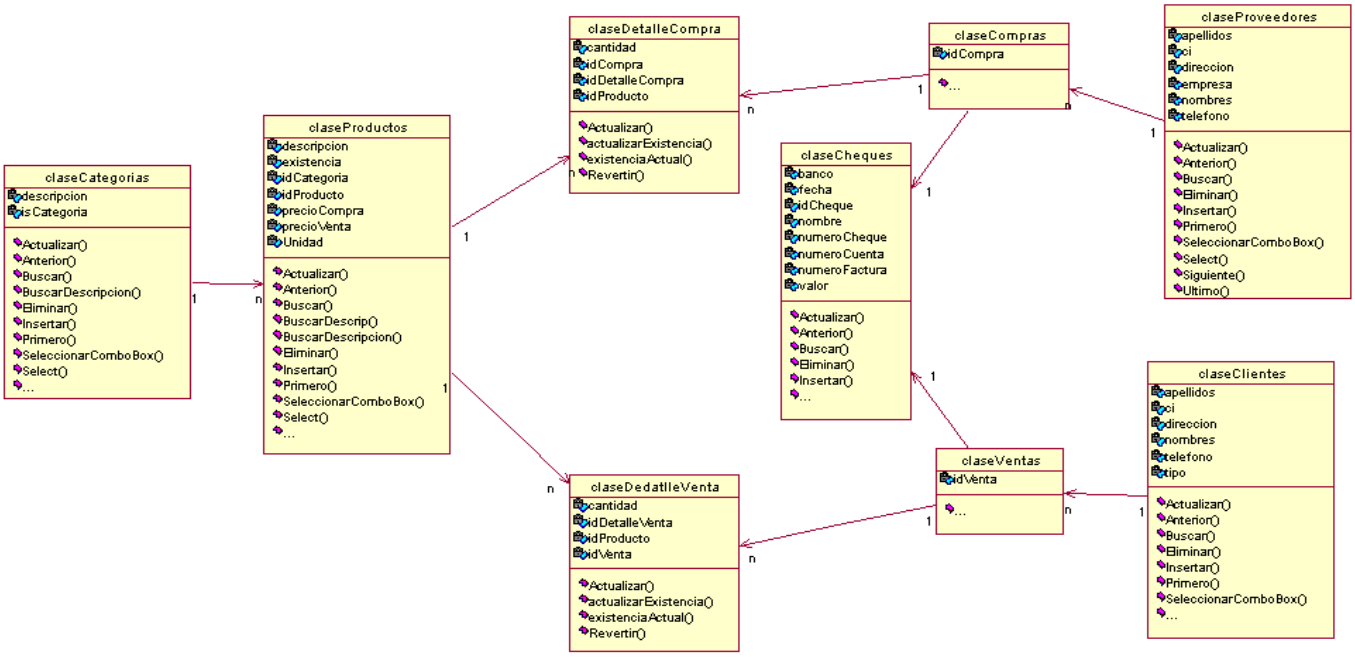


Figura16. Diagrama de Clases 1

6.6.1.4.3 Diagrama de Secuencia

Es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.[9]

[8] http://es.wikipedia.org/wiki/Diagrama_de_clases

[9] http://es.wikipedia.org/wiki/Diagrama_de_secuencia

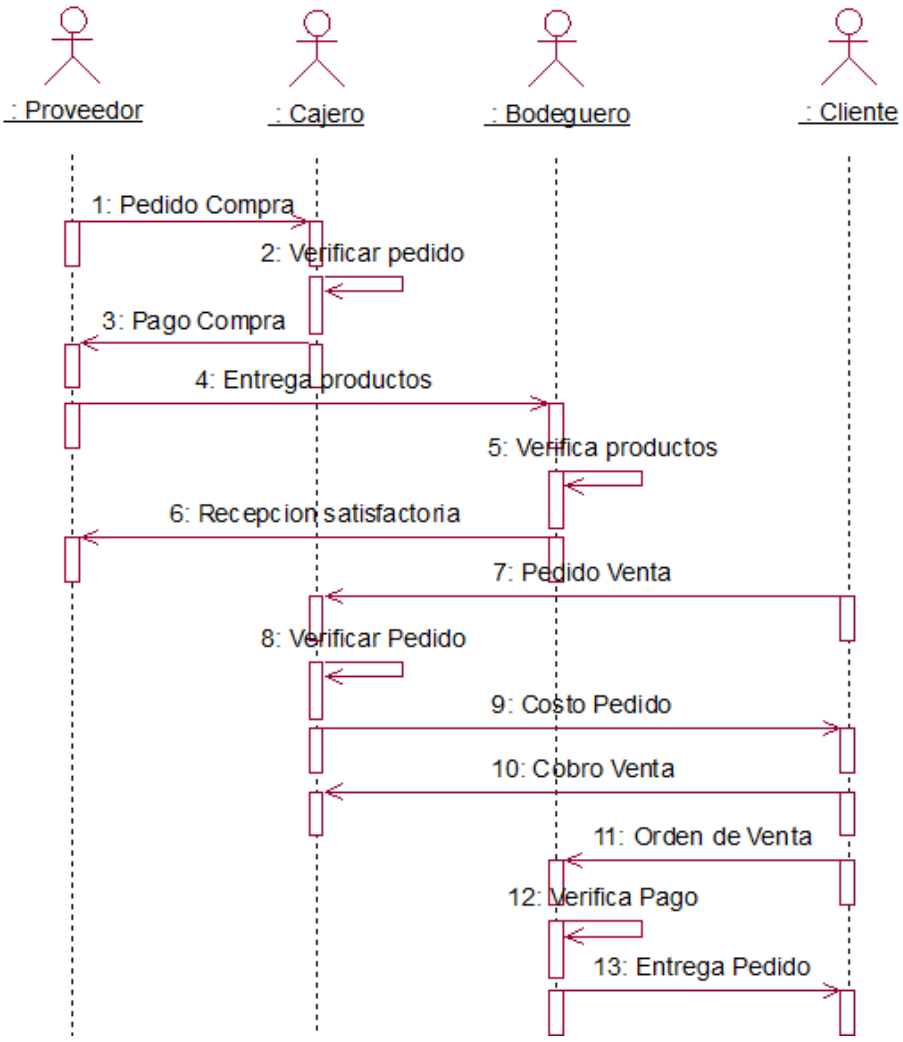


Figura 17. Diagrama de Secuencia 1

6.6.1.4.4 Diagrama Colaboración

Muestra cómo las instancias específicas de las clases trabajan juntas para conseguir un objetivo común.

Implementa las asociaciones del diagrama de clases mediante el paso de mensajes de un objeto a otro. Dicha implementación es llamada "enlace". [10]

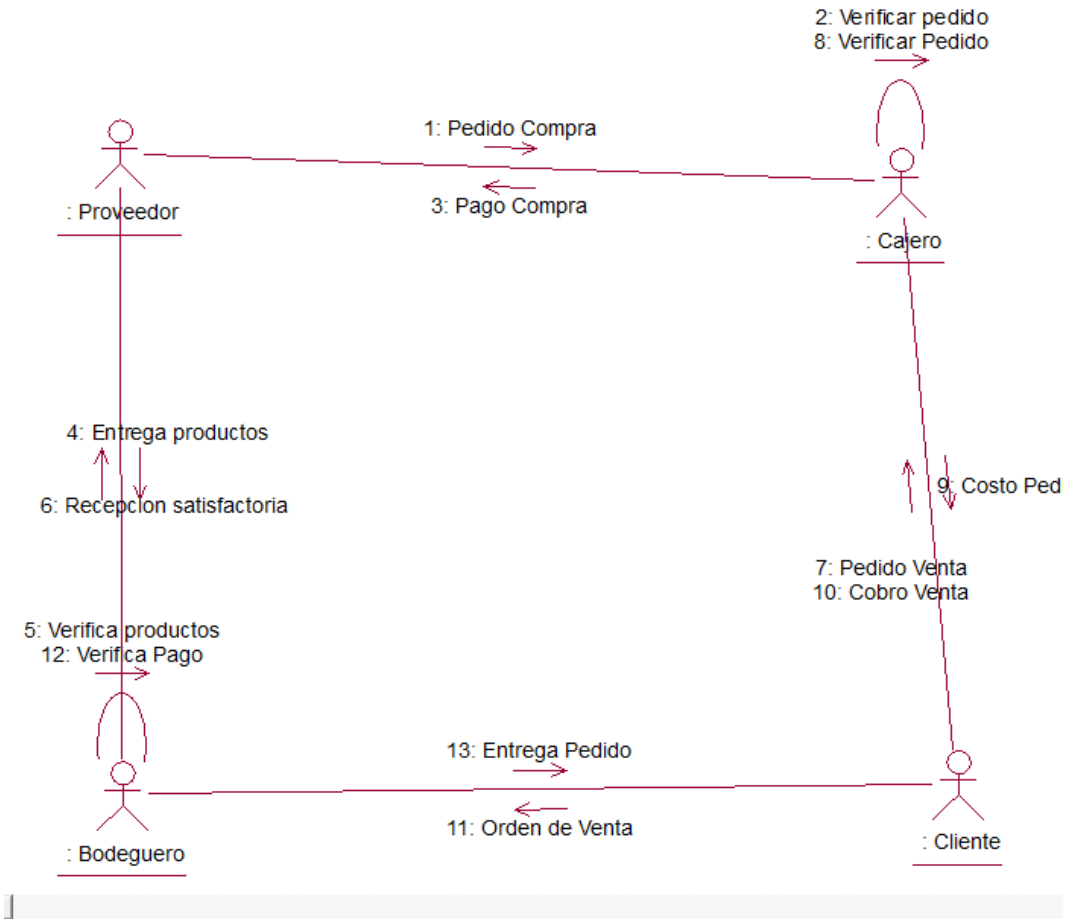


Figura 18. Diagrama de Colaboración 1

6.6.2 Diseño del sistema

6.6.2.1 Diseño de la Base de Datos

[10] http://es.wikipedia.org/wiki/Diagrama_de_colaboraci%C3%B3n

La Ferretería Bolivariana cuenta con una base de datos que almacena el stock de los productos de la empresa para poder saber cuáles son los que se tienen existentes y cuales están en faltantes.

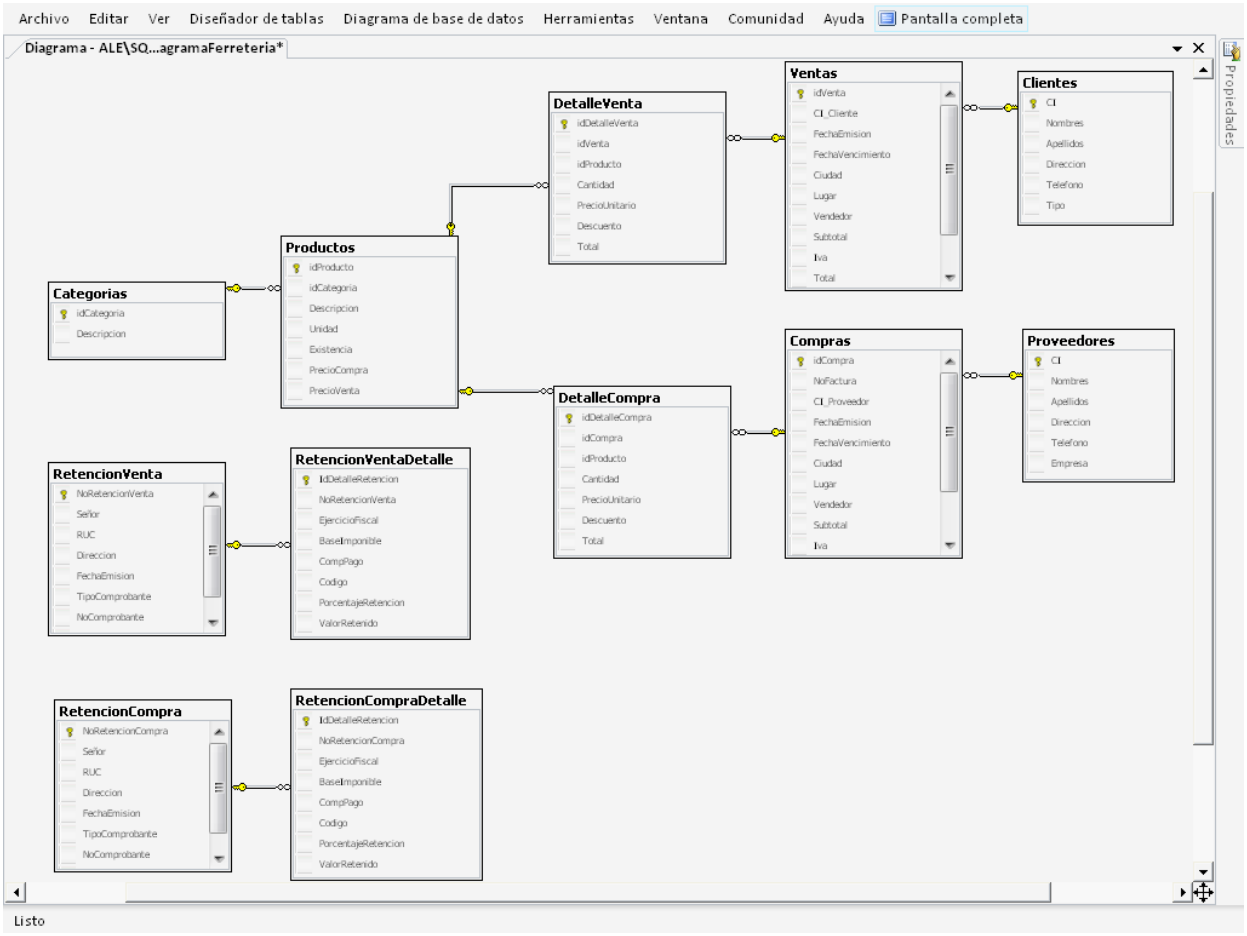


Figura19. Base de Datos 1

6.6.2.2 Diccionario de datos

Diccionario de Datos de la Base de Datos Ferretería

No.	Nombre	Tipo	Descripción
1	idCategoria	Int	Es el código de la

			categoría
2	Descripcion	nvarchar(100)	nombre de la categoría o clasificación de productos

Tabla 6. CATEGORIA

No.	**Nombre**	**Tipo**	**Descripción**
1	idProducto	int	Almacena el código del producto
2	idCategoria	int	Guarda la categoría a la que pertenece el producto
3	Descripcion	nvarchar(150)	Nombre del Producto
4	Unidad	nvarchar(10)	Es la unidad de medida del producto
5	Existencia	int	Es el numero de productos existentes
6	PrecioCompra	float	Precio de compra del producto
7	PrecioVenta	float	Precio de venta del Producto

Tabla7. PRODUCTOS

No.	**Nombre**	**Tipo**	**Descripción**

1	CI	nvarchar(10)	Almacena la cedula de identidad del proveedor
2	Nombres	nvarchar(50)	Nombres del proveedor
3	Apellidos	nvarchar(50)	Apellidos del proveedor
4	Direccion	nvarchar(100)	Dirección del proveedor
5	Telefono	nvarchar(10)	Teléfono celular o convencional del proveedor
6	Empresa	nvarchar(50)	Nombre de la empresa distribuidora

Tabla8. PROVEEDORES

No.	**Nombre**	**Tipo**	**Descripción**
1	CI	nvarchar(10)	Almacena la cedula de identidad del cliente
2	Nombres	nvarchar(50)	Nombres del cliente
3	Apellidos	nvarchar(50)	Apellidos del cliente
4	Direccion	nvarchar(100)	Dirección del cliente
5	Telefono	nvarchar(12)	Teléfono celular o convencional del cliente

6	Tipo	nvarchar(12)	Tipo de cliente natural o contribuyente especial.

Tabla9. CLIENTES

No.	**Nombre**	**Tipo**	**Descripción**
1	IdCheque	int	Es el código que asigna el sistema al cheque
2	NumeroFactura	nchar(10)	Es el numero de factura que un cliente nos cancelan o cancelamos al proveedor por medio del cheque.
3	Nombre	varchar(50)	Almacena el nombre y apellido del dueño del cheque
4	Banco	nvarchar(100)	El banco al que pertenece el cheque
5	NumeroCuenta	varchar(50)	El número de cuenta del cheque
6	NumeroCheque	varchar(50)	El numero de cheque
7	Fecha	Datetime	La fecha del cheque
8	Valor	float	La cantidad en dólares del cheque

Tabla 10. CHEQUES

No.	Nombre	Tipo	Descripción
1	IdCompra	int	Es el código que asigna el sistema a la factura de compra.
2	NoFactura	nchar(10)	Es el número real de la factura de compra
3	CI_Proveedor	nchar(10)	Almacena la cedula del proveedor pero muestra su nombre.
4	FechaEmision	datetime	La fecha de la factura de compra
5	FechaVencimiento	datetime	La fecha en que vence el pago de la factura.
6	Ciudad	nvarchar(20)	La ciudad a la que pertenece el proveedor
7	Lugar	nvarchar(20)	La dirección del proveedor
8	Vendedor	nvarchar(30)	Es el nombre del vendedor que realizo el pedido.
9	Subtotal	decimal(18,2)	Es la sumatoria de los totales del detalle de la factura de compra.
10	Iva	decimal(18,2)	Es el 12% del subtotal de la factura.
11	Total	decimal(18,2)	Es la sumatoria del subtotal más el IVA.
12	Facturado	Bit	Este campo se pone un visto si esta facturado o actualizado las existencias caso contrario solo aparecerá el cuadro

			vacio del check.

Tabla 11. COMPRAS

No.	Nombre	Tipo	Descripción
1	IdDetalleCompra	int	Es el código que asigna el sistema al detalle de la factura de compra.
2	IdCompra	int	Es el código de la factura a la que pertenece
3	IdProducto	int	Es el código del producto
4	Cantidad	Int	La cantidad de productos
5	PrecioUnitario	float	El precio unitario de costo
6	Descuento	float	El descuento que recibe el producto
7	Total	float	La cantidad por el valor unitario

Tabla 12. DETALLECOMPRA

No.	Nombre	Tipo	Descripción
1	IdVenta	nchar(10)	Es el número real de la factura de venta
3	CI_Cliente	nchar(10)	Almacena la cedula cliente pero muestra su nombre.

4	FechaEmision	Datetime	La fecha de la factura de venta
5	FechaVencimiento	Datetime	La fecha en que vence el pago de la factura.
6	Ciudad	nvarchar(20)	La ciudad a la que pertenece el proveedor
7	FormadePago	nvarchar(20)	Se escoge si es efectivo o crédito
8	Vendedor	nvarchar(30)	Es el nombre del vendedor que realizo el pedido.
9	Subtotal	decimal(18,2)	Es la sumatoria de los totales del detalle de la factura de venta.
10	Iva	decimal(18,2)	Es el 12% del subtotal de la factura.
11	Total	decimal(18,2)	Es la sumatoria del subtotal más el IVA.
12	Facturado	Bit	Este campo se pone un visto si esta facturado o actualizado las existencias caso contrario solo aparecerá el cuadro vacio del check.

Tabla 13. VENTAS

No.	**Nombre**	**Tipo**	**Descripción**
1	IdDetalleVenta	int	Es el código que asigna el sistema al detalle de la factura de venta.

2	IdVenta	int	Es el código de la factura a la que pertenece
3	IdProducto	int	Es el código del producto
4	Cantidad	Int	La cantidad de productos
5	PrecioUnitario	float	El precio unitario de costo
6	Descuento	float	El descuento que recibe el producto
7	Total	float	La cantidad por el valor unitario

Tabla 14. DETALLEVENTAS

No.	**Nombre**	**Tipo**	**Descripción**
1	NoRetencionCompra	nchar(10)	Es el número real de la retención emitida por la ferretería bolivariana.
2	Señor	varchar(50)	Es el nombre del señor o la empresa a la que va dirigida.
3	RUC	varchar(50)	Es la cedula o ruc de nuestro proveedor
4	Direccion	varchar(50)	Es la dirección de nuestro proveedor
5	FechaEmision	datetime	Es la fecha en la que se realiza la retención
6	TipoComprobante	varchar(50)	Es el tipo de documento de compra-

			venta
7	NoComprobante	varchar(50)	Es el número del documento de compra
8	Total	decimal(18,2)	Es la suma de los totales del detalle de la retención.

Tabla 15. RETENCIONCOMPRA

No.	**Nombre**	**Tipo**	**Descripción**
1	IdDetalleRetencion	int	Es el código que el sistema asigna al detalle de la retención.
2	NoRetencionCompra	nchar(10)	Es a la retención compra que pertenece el detalle.
3	EjercicioFiscal	varchar(50)	Es el año en que se realiza la retención
4	BaseImponible	float	Es el subtotal de la factura de compra
5	CompPago	varchar(50)	Es el número de comprobante de pago
6	Codigo	nchar(10)	Es el código del impuesto a retener
7	PorcentajeRetencion	varchar(50)	Es el porcentaje que se retiene de la base imponible según el código del impuesto
8	ValorRetenido	float	Es el porcentaje de la base imponible calculado

Tabla 16. RETENCIONCOMPRADETALLE

No.	Nombre	Tipo	Descripción
1	NoRetencionVenta	nchar(10)	Es el número del comprobante de retención que recibimos de nuestro cliente.
2	Señor	varchar(50)	Es el nombre del señor o la empresa de nuestro cliente
3	RUC	varchar(50)	Es la cedula o ruc de nuestro cliente
4	Direccion	varchar(50)	Es la dirección de nuestro cliente
5	FechaEmision	datetime	Es la fecha en la que se realiza la retención
6	TipoComprobante	varchar(50)	Es el tipo de documento de venta
7	NoComprobante	varchar(50)	Es el número de la factura de venta
8	Total	decimal(18,2)	Es la suma de los totales del detalle de la retención.

Tabla 17. RETENCIONVENTA

No.	Nombre	Tipo	Descripción
1	IdDetalleRetencion	int	Es el código que el sistema asigna al detalle de la retención.
2	NoRetencionCompra	nchar(10)	Es a la retención compra que pertenece el

			detalle.
3	EjercicioFiscal	varchar(50)	Es el año en que se realiza la retención
4	BaseImponible	float	Es el subtotal de la factura de venta
5	CompPago	varchar(50)	Es el número de comprobante de pago
6	Codigo	nchar(10)	Es el código del impuesto a retener
7	PorcentajeRetencion	varchar(50)	Es el porcentaje que se retiene de la base imponible según el código del impuesto
8	ValorRetenido	Float	Es el porcentaje de la base imponible calculado

Tabla 18. RETENCIONVENTADETALLE

6.6.2.3 Diseño de Interfaces

6.6.2.3.1 Diseño de Entradas

Se utilizo un estándar en los formularios de ingresos, todos tienen botones de Agregar (Ingresar Registro), Editar (Editar Información), Eliminar (Eliminar Registro), Guardar (Guardar Registro), Cancelar (Cancelar la Acción), Cerrar (Cerrar el Formulario); también cada formulario tiene un navegador para poder encontrar registros.

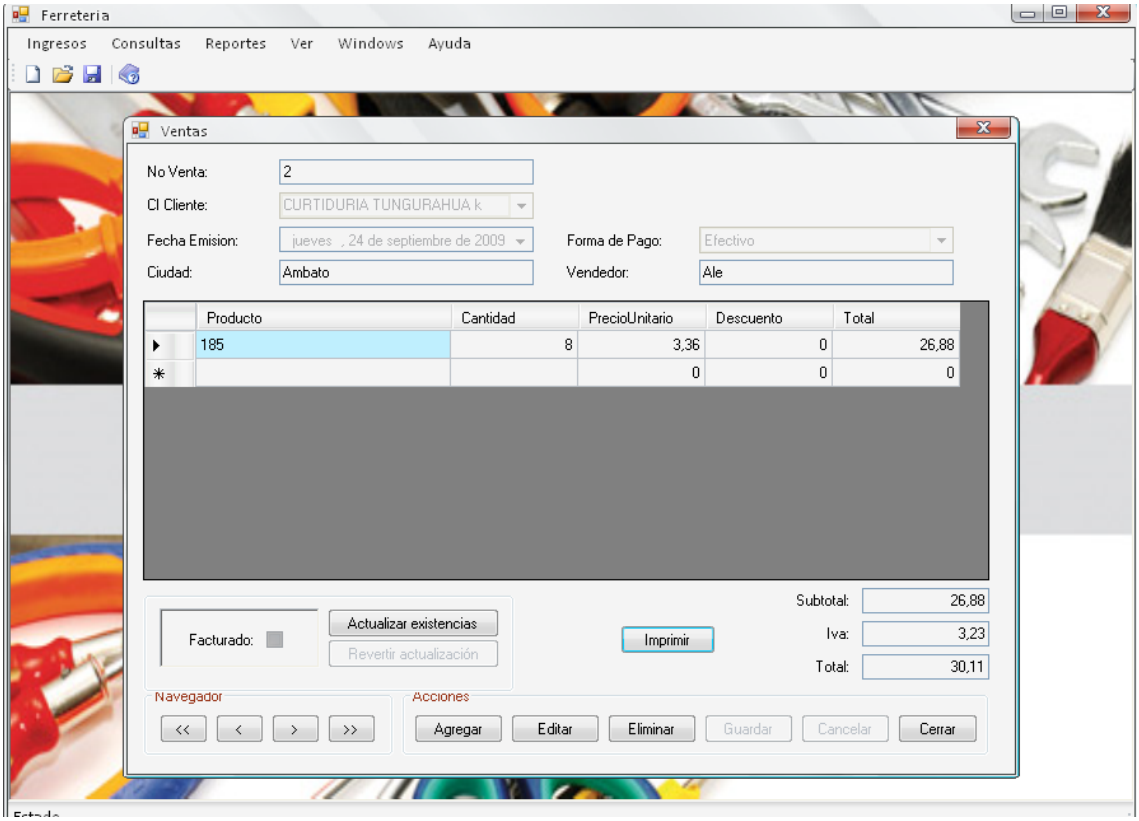


Figura 20. Diseño de Entrada 1

6.6.2.3.2 Diseño de Salidas

Los formularios de consultas o diseños de salidas están basados en consultas por medio de filtraciones alfabéticas o llamadas búsquedas inteligentes.

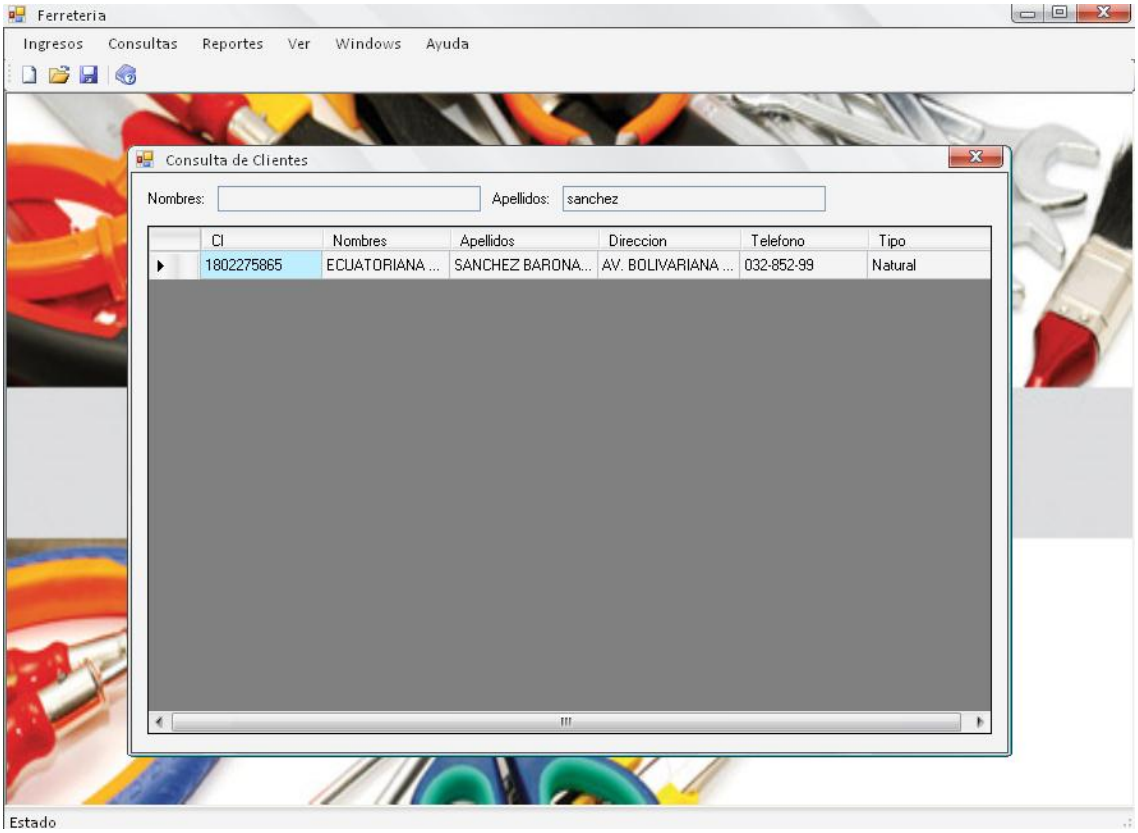


Figura 21. Diseño de Salidas 1

6.6.2.3.3 Reportes

Los reportes tienen un formato que permite exportar informe, actualizar, imprimir el reporte, maximizar, minimizar y navegar en los registros.

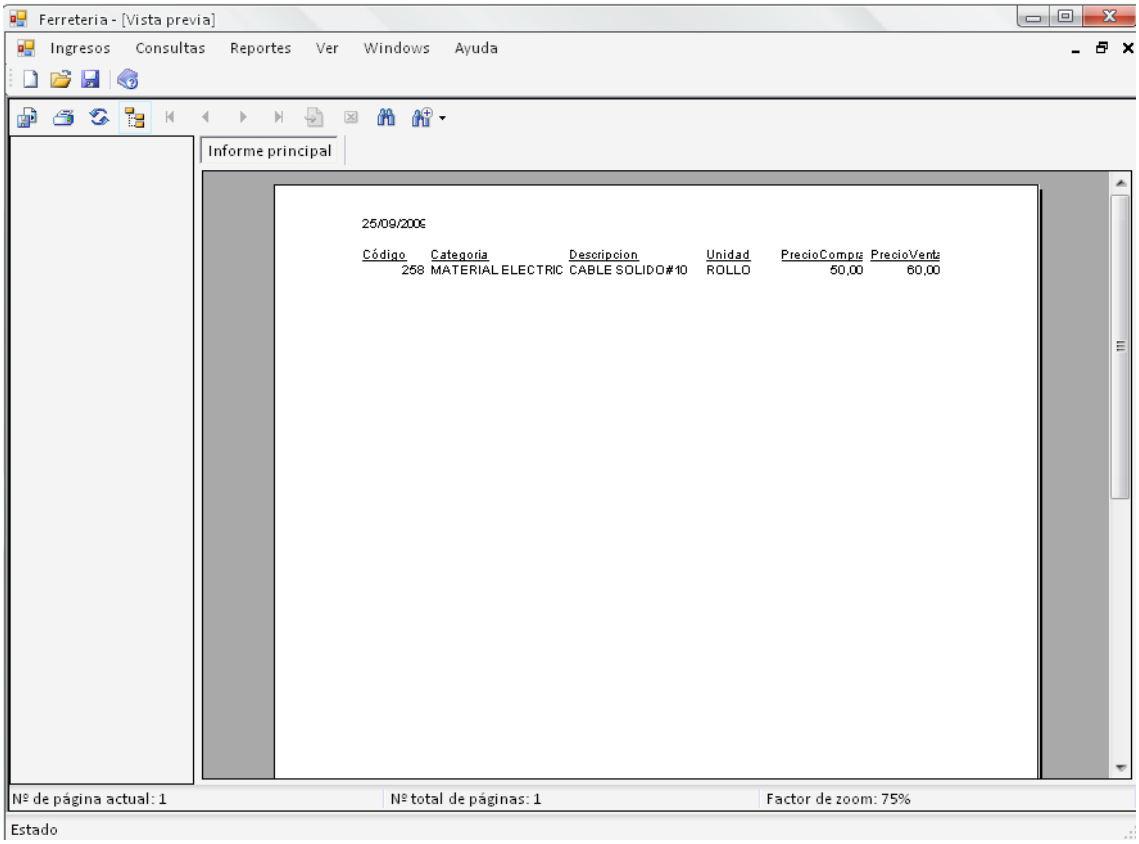


Figura 22. Reportes 1

6.6.3 Codificación

6.6.3.1 Código Base del Sistema

Clase Productos

```
using System;

using System.Collections.Generic;

using System.Text;

using System.Data;

using System.Windows.Forms;
```

```
using System.Data.SqlClient;


namespace Ferreteria
{
//creamos la clase mediante propiedades
    class claseProductos
    {
        private string idProducto;
        private string idCategoria;
        private string descripcion;
        private string unidad;
        private string existencia;
        private string precioCompra;
        private string precioVenta;


        public string IdProducto
        {
            get
            {
                return idProducto;
            }
```

```
    set
    {
        idProducto = value;
    }
}

public string IdCategoria
{
    get
    {
        return idCategoria;
    }
    set
    {
        idCategoria = value;
    }
}

public string Descripcion
{
    get
```

```
    {
        return descripcion;
    }
    set
    {
        descripcion = value;
    }
}

public string Unidad
{
    get
    {
        return unidad;
    }
    set
    {
        unidad = value;
    }
}
```

```
public string Existencia
{
    get
    {
        return existencia;
    }
    set
    {
        existencia = value;
    }
}

public string PrecioCompra
{
    get
    {
        return precioCompra;
    }
    set
    {
        precioCompra = value;
```

```
    }
}

public string PrecioVenta
{
    get
    {
        return precioVenta;
    }
    set
    {
        precioVenta = value;
    }
}
```

//Tenemos el primer metodo de la clase productos que nos servira para insertar productos nuevos.

```
public void Insertar()
{
```

// **En primer lugar se crea la conexión con la base de datos mediente una cadena de conexión que usa usuario y contraseña)**

```
    SqlConnection conexion = new SqlConnection();
```

```
try
{
    conexion.ConnectionString = claseConexion.Conexion;
```

//se crea una sentenciaInsert mediante codigo sql que me permita insertar un nuevo producto con la instruccion INSERT INTO, en la cual enviamos todos los campos de la tabla productos a las variables de la base de datos.

```
    string sentenciaInsert = "INSERT INTO productos " +
                "(IdCategoria, Descripcion, Unidad, Existencia, PrecioCompra, PrecioVenta) " +
                "VALUES (@vIdCategoria, @vDescripcion, @vUnidad, @vExistencia, @vPrecioCompra, @vPrecioVenta)";

    SqlCommand cmdInsert = new SqlCommand();
    cmdInsert.CommandText = sentenciaInsert;
    cmdInsert.CommandType = CommandType.Text;
    cmdInsert.Connection = conexion;
```

//se especica los parametros

```
    SqlParameter parametro2 = new SqlParameter("@vIdCategoria", SqlDbType.NChar, 6);
    SqlParameter parametro3 = new SqlParameter("@vDescripcion", SqlDbType.NVarChar, 150);
    SqlParameter parametro4 = new SqlParameter("@vUnidad", SqlDbType.NVarChar, 10);
    SqlParameter parametro5 = new SqlParameter("@vExistencia", SqlDbType.Int);
```

SqlParameter parametro6 = new SqlParameter("@vPrecioCompra", SqlDbType.Float);

SqlParameter parametro7 = new SqlParameter("@vPrecioVenta", SqlDbType.Float);

// **Se agrega los valores a cada uno de los parámetros**

cmdInsert.Parameters.Add(parametro2);

cmdInsert.Parameters.Add(parametro3);

cmdInsert.Parameters.Add(parametro4);

cmdInsert.Parameters.Add(parametro5);

cmdInsert.Parameters.Add(parametro6);

cmdInsert.Parameters.Add(parametro7);

cmdInsert.Parameters["@vIdCategoria"].Value = idCategoria;

cmdInsert.Parameters["@vDescripcion"].Value = descripcion;

cmdInsert.Parameters["@vUnidad"].Value = unidad;

cmdInsert.Parameters["@vExistencia"].Value = existencia;

cmdInsert.Parameters["@vPrecioCompra"].Value = precioCompra;

cmdInsert.Parameters["@vPrecioVenta"].Value = precioVenta;

conexion.Open();

// **Ejecuta la instrucción sql**

```
            cmdInsert.ExecuteNonQuery();
        }
        catch (SqlException error)
        {
            claseExcepciones.Windows(error);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message, "Error C#");
        }
        finally
        {
            conexion.Close();
        }
    }
```

//Metodo Eliminar, nos servira para eliminar un producto

```
    public void Eliminar()
    {
        try
        {
```

```
//Conexion a la base de datos
        SqlConnection conexion = new SqlConnection(claseConexion.Conexion);
```

//creacion de la sentenciaDelete que nos permite borrar el producto que tenga el idProducto o codigo que se le envia del sistema mediante la instrucción DELETE

```
        string sentenciaDelete = "DELETE FROM Productos " +
                                 "WHERE IdProducto = @vIdProducto";


        SqlCommand cmdDelete = new SqlCommand(sentenciaDelete, conexion);
        cmdDelete.Parameters.Add("@vIdProducto", SqlDbType.NChar, 6);
        cmdDelete.Parameters["@vIdProducto"].Value = idProducto;
        conexion.Open();
        cmdDelete.ExecuteNonQuery();
        conexion.Close();
        conexion.Dispose();
      }
      catch (SqlException error)
      {
        claseExcepciones.Windows(error);
      }
      catch (Exception error)
      {
        MessageBox.Show(error.Message, "Error C#");
```

```
        }
    }
```

//Metodo Actualizar, nos sirve para actualizar un producto mediante la instrucción UPDATE SET

```
    public void Actualizar()
    {
```

//Conexion a la base de datos

```
        SqlConnection conexion = new SqlConnection();
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;
//creacion de la sentenciaUpdate que me permitira actualizar el
            string sentenciaUpdate = "UPDATE Productos " +
                                "SET IdCategoria = @vIdCategoria, Descripcion = @vDescripcion, Unidad = @vUnidad, Existencia = @vExistencia, PrecioCompra = @vPrecioCompra, PrecioVenta = @vPrecioVenta " +
                                "WHERE IdProducto = @vIdProducto";

            SqlCommand cmdInsert = new SqlCommand();
            cmdInsert.CommandText = sentenciaUpdate;
            cmdInsert.CommandType = CommandType.Text;
```

```
cmdInsert.Connection = conexion;


SqlParameter parametro1 = new SqlParameter("@vIdProducto",
SqlDbType.NChar, 6);

SqlParameter parametro2 = new SqlParameter("@vIdCategoria",
SqlDbType.NChar, 6);

SqlParameter parametro3 = new SqlParameter("@vDescripcion",
SqlDbType.NVarChar, 150);

SqlParameter parametro4 = new SqlParameter("@vUnidad",
SqlDbType.NVarChar, 10);

SqlParameter parametro5 = new SqlParameter("@vExistencia", SqlDbType.Int);

SqlParameter parametro6 = new SqlParameter("@vPrecioCompra",
SqlDbType.Float);

SqlParameter parametro7 = new SqlParameter("@vPrecioVenta",
SqlDbType.Float);


cmdInsert.Parameters.Add(parametro1);

cmdInsert.Parameters.Add(parametro2);

cmdInsert.Parameters.Add(parametro3);

cmdInsert.Parameters.Add(parametro4);

cmdInsert.Parameters.Add(parametro5);

cmdInsert.Parameters.Add(parametro6);

cmdInsert.Parameters.Add(parametro7);
```

```
    cmdInsert.Parameters["@vIdProducto"].Value = idProducto;
    cmdInsert.Parameters["@vIdCategoria"].Value = idCategoria;
    cmdInsert.Parameters["@vDescripcion"].Value = descripcion;
    cmdInsert.Parameters["@vUnidad"].Value = unidad;
    cmdInsert.Parameters["@vExistencia"].Value = existencia;
    cmdInsert.Parameters["@vPrecioCompra"].Value = precioCompra;
    cmdInsert.Parameters["@vPrecioVenta"].Value = precioVenta;

    conexion.Open();
    cmdInsert.ExecuteNonQuery();
}
catch (SqlException error)
{
    claseExcepciones.Windows(error);
}
catch (Exception error)
{
    MessageBox.Show(error.Message, "Error C#");
}
finally
```

```
        {
            conexion.Close();
        }
    }
```

//Metodo Buscar.- busca el producto por el código

```
    public void Buscar(string codigo)
    {
        SqlConnection conexion = new SqlConnection();
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;
            string sentenciaSelect = "SELECT * FROM Productos";
            string tempCodigo;

            SqlCommand cmdSelect = new SqlCommand(sentenciaSelect, conexion);
            cmdSelect.CommandText = sentenciaSelect;
            cmdSelect.Connection = conexion;
            conexion.Open();

            SqlDataReader drBuscar = cmdSelect.ExecuteReader();
            idProducto = "-";
```

```
    while (drBuscar.Read())
    {
        tempCodigo = Convert.ToString(drBuscar["IdProducto"]).Trim();
        if (tempCodigo == codigo)
        {
            idProducto = Convert.ToString(drBuscar["IdProducto"]);
            idCategoria = Convert.ToString(drBuscar["IdCategoria"]);
            descripcion = Convert.ToString(drBuscar["Descripcion"]);
            unidad = Convert.ToString(drBuscar["Unidad"]);
            existencia = Convert.ToString(drBuscar["Existencia"]);
            precioCompra = Convert.ToString(drBuscar["PrecioCompra"]);
            precioVenta = Convert.ToString(drBuscar["PrecioVenta"]); ;
        }
    }
    drBuscar.Close();
    conexion.Close();
}
catch (SqlException error)
{
    claseExcepciones.Windows(error);
}
```

```
        catch (Exception error)
        {
            MessageBox.Show(error.Message, "Error C#");
        }
        finally
        {
            conexion.Close();
            conexion.Dispose();
        }
    }
```

//Buscar descripción.- este método utilizamos para buscar descripción del producto y no ingresar productos duplicados.

```
    public void BuscarDescripcion(string buscarDescripcion)
    {
        SqlConnection conexion = new SqlConnection();
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;
            string sentenciaSelect = "SELECT * FROM productos " +
                                     "WHERE Descripcion LIKE '%" + buscarDescripcion + "%'";
```

```
    SqlCommand cmdSelect = new SqlCommand(sentenciaSelect, conexion);
    cmdSelect.CommandText = sentenciaSelect;
    cmdSelect.Connection = conexion;
    conexion.Open();


    SqlDataReader drBuscar = cmdSelect.ExecuteReader();
    idProducto = "-";
    if (drBuscar.Read())
    {
        idProducto = Convert.ToString(drBuscar["IdProducto"]);
        idCategoria = Convert.ToString(drBuscar["IdCategoria"]);
        descripcion = Convert.ToString(drBuscar["Descripcion"]);
        unidad = Convert.ToString(drBuscar["Unidad"]);
        existencia = Convert.ToString(drBuscar["Existencia"]);
        precioCompra = Convert.ToString(drBuscar["PrecioCompra"]);
        precioVenta = Convert.ToString(drBuscar["PrecioVenta"]);
        MessageBox.Show(descripcion);
    }
    drBuscar.Close();
    conexion.Close();
}
```

```
        catch (SqlException error)
        {
            claseExcepciones.Windows(error);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message, "Error C#");
        }
        finally
        {
            conexion.Close();
            conexion.Dispose();
        }
    }
```

// Buscar descrip se utiliza para realizar la buscada por medio de la descripción del producto

```
    public void BuscarDescrip(string buscarDescripcion)
    {
        SqlConnection conexion = new SqlConnection();
        try
        {
```

```
conexion.ConnectionString = claseConexion.Conexion;
string sentenciaSelect = "SELECT * FROM productos " +
                         "WHERE Descripcion LIKE '" + buscarDescripcion + "%'";

SqlCommand cmdSelect = new SqlCommand(sentenciaSelect, conexion);
cmdSelect.CommandText = sentenciaSelect;
cmdSelect.Connection = conexion;
conexion.Open();

SqlDataReader drBuscar = cmdSelect.ExecuteReader();
idProducto = "-";
if (drBuscar.Read())
{
   idProducto = Convert.ToString(drBuscar["IdProducto"]);
   idCategoria = Convert.ToString(drBuscar["IdCategoria"]);
   descripcion = Convert.ToString(drBuscar["Descripcion"]);
   unidad = Convert.ToString(drBuscar["Unidad"]);
   existencia = Convert.ToString(drBuscar["Existencia"]);
   precioCompra = Convert.ToString(drBuscar["PrecioCompra"]);
   precioVenta = Convert.ToString(drBuscar["PrecioVenta"]);
}
```

```
            drBuscar.Close();
            conexion.Close();
        }
        catch (SqlException error)
        {
            claseExcepciones.Windows(error);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message, "Error C#");
        }
        finally
        {
            conexion.Close();
            conexion.Dispose();
        }
    }
```

//Este metodo me llena un combo box con la lista de productos

```
    public DataTable SeleccionarComboBox()
    {
        SqlConnection conexion = new SqlConnection();
```

```
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;

            StringBuilder sbSentenciaSelect = new StringBuilder();
            sbSentenciaSelect.Append("SELECT IdProducto, Descripcion ");
            sbSentenciaSelect.Append("FROM Productos");

            SqlCommand cmdSelect = new SqlCommand(sbSentenciaSelect.ToString(),
conexion);
            conexion.Open();

            SqlDataReader drProductos = cmdSelect.ExecuteReader();
            DataTable dtProductos = new DataTable();
            dtProductos.Load(drProductos);
            return dtProductos;
        }
        catch (SqlException error)
        {
            throw error;
        }
        catch (Exception error)
```

```
        {
            throw error;
        }
        finally
        {
            //throw error;
        }
    }
```

//Este metodo hace que mediante el navegador se muestre el primer registro de la tabla productos.

```
    public void Primero()
    {
        SqlConnection conexion = new SqlConnection();
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;
            string sentenciaSelect = "SELECT * FROM Productos";

            SqlCommand cmdSelect = new SqlCommand(sentenciaSelect, conexion);
            cmdSelect.CommandText = sentenciaSelect;
            cmdSelect.Connection = conexion;
            conexion.Open();
```

```
    SqlDataReader drBuscar = cmdSelect.ExecuteReader();
    idProducto = "-";
    if (drBuscar.Read())
    {
        idProducto = Convert.ToString(drBuscar["IdProducto"]);
        idCategoria = Convert.ToString(drBuscar["IdCategoria"]);
        descripcion = Convert.ToString(drBuscar["Descripcion"]);
        unidad = Convert.ToString(drBuscar["Unidad"]);
        existencia = Convert.ToString(drBuscar["Existencia"]);
        precioCompra = Convert.ToString(drBuscar["PrecioCompra"]);
        precioVenta = Convert.ToString(drBuscar["PrecioVenta"]);
    }
    drBuscar.Close();
    conexion.Close();
}
catch (SqlException error)
{
    claseExcepciones.Windows(error);
}
catch (Exception error)
```

```
        {
            MessageBox.Show(error.Message, "Error C#");
        }
        finally
        {
            conexion.Close();
            conexion.Dispose();
        }
    }
```

//Este metodo hace que mediante el navegador se muestre el anterior registro del que esta seleccionado en la tabla productos.

```
    public void Anterior(string codigo)
    {
        SqlConnection conexion = new SqlConnection();
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;
            string sentenciaSelect = "SELECT * FROM Productos";
            string tempCodigo;
```

```
SqlCommand cmdSelect = new SqlCommand(sentenciaSelect, conexion);
cmdSelect.CommandText = sentenciaSelect;
cmdSelect.Connection = conexion;
conexion.Open();

SqlDataReader drBuscar = cmdSelect.ExecuteReader();
idProducto = "-";
while (drBuscar.Read())
{
   tempCodigo = Convert.ToString(drBuscar["IdProducto"]).Trim();
   if (codigo == "")
   {
      //primer registro
      Primero();
      break;
   }
   if (tempCodigo == codigo)
   {
      //Registro encontrado
      break;
   }
```

```
        else
        {
            idProducto = Convert.ToString(drBuscar["IdProducto"]);
            idCategoria = Convert.ToString(drBuscar["IdCategoria"]);
            descripcion = Convert.ToString(drBuscar["Descripcion"]);
            unidad = Convert.ToString(drBuscar["Unidad"]);
            existencia = Convert.ToString(drBuscar["Existencia"]);
            precioCompra = Convert.ToString(drBuscar["PrecioCompra"]);
            precioVenta = Convert.ToString(drBuscar["PrecioVenta"]);
        }
    }
    drBuscar.Close();
    conexion.Close();
}
catch (SqlException error)
{
    claseExcepciones.Windows(error);
}
catch (Exception error)
{
    MessageBox.Show(error.Message, "Error C#");
```

```
    }
    finally
    {
        conexion.Close();
        conexion.Dispose();
    }
}
```

//Este metodo hace que mediante el navegador se muestre el siguiente registro del seleccionado en la tabla productos.

```
public void Siguiente(string codigo)
{
    SqlConnection conexion = new SqlConnection();
    try
    {
        conexion.ConnectionString = claseConexion.Conexion;
        string sentenciaSelect = "SELECT * FROM Productos";
        string tempCodigo;

        SqlCommand cmdSelect = new SqlCommand(sentenciaSelect, conexion);
        cmdSelect.CommandText = sentenciaSelect;
```

```
cmdSelect.Connection = conexion;
conexion.Open();

SqlDataReader drBuscar = cmdSelect.ExecuteReader();
idProducto = "-";
while (drBuscar.Read())
{
   if (codigo == "")
   {
      //ultimo registro
      Ultimo();
      break;
   }
   tempCodigo = Convert.ToString(drBuscar["IdProducto"]).Trim();
   if (tempCodigo == codigo)
   {
      if (drBuscar.Read())
      {
         idProducto = Convert.ToString(drBuscar["IdProducto"]);
         idCategoria = Convert.ToString(drBuscar["IdCategoria"]);
         descripcion = Convert.ToString(drBuscar["Descripcion"]);
```

```
                unidad = Convert.ToString(drBuscar["Unidad"]);
                existencia = Convert.ToString(drBuscar["Existencia"]);
                precioCompra = Convert.ToString(drBuscar["PrecioCompra"]);
                precioVenta = Convert.ToString(drBuscar["PrecioVenta"]);
            }
            break;
        }
    }
    drBuscar.Close();
    conexion.Close();
}
catch (SqlException error)
{
    claseExcepciones.Windows(error);
}
catch (Exception error)
{
    MessageBox.Show(error.Message, "Error C#");
}
finally
{
```

```
            conexion.Close();
            conexion.Dispose();
        }
    }
```

//Este metodo hace que mediante el navegador se muestre el ultimo registro de la tabla productos.

```
    public void Ultimo()
    {
        SqlConnection conexion = new SqlConnection();
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;
            string sentenciaSelect = "SELECT * FROM Productos";

            SqlCommand cmdSelect = new SqlCommand(sentenciaSelect, conexion);
            cmdSelect.CommandText = sentenciaSelect;
            cmdSelect.Connection = conexion;
            conexion.Open();

            SqlDataReader drBuscar = cmdSelect.ExecuteReader();
            idProducto = "-";
```

```
    while (drBuscar.Read())
    {
        idProducto = Convert.ToString(drBuscar["IdProducto"]);
        idCategoria = Convert.ToString(drBuscar["IdCategoria"]);
        descripcion = Convert.ToString(drBuscar["Descripcion"]);
        unidad = Convert.ToString(drBuscar["Unidad"]);
        existencia = Convert.ToString(drBuscar["Existencia"]);
        precioCompra = Convert.ToString(drBuscar["PrecioCompra"]);
        precioVenta = Convert.ToString(drBuscar["PrecioVenta"]);
    }
    drBuscar.Close();
    conexion.Close();
}
catch (SqlException error)
{
    claseExcepciones.Windows(error);
}
catch (Exception error)
{
    MessageBox.Show(error.Message, "Error C#");
}
```

```
        finally
        {
            conexion.Close();
            conexion.Dispose();
        }
    }
```

//Select.- Permite obtener una lista de los productos utilizando una descripción está basado en un procedimiento almacenado

```
    public DataSet Select(string spAlmacenado, string des)
    {
        SqlConnection conexion = new SqlConnection();
        try
        {
            conexion.ConnectionString = claseConexion.Conexion;
            //Segunda forma.
            SqlCommand spConsulta = new SqlCommand(spAlmacenado, conexion);
            //spConsulta.CommandType = CommandType.StoredProcedure;

            //Declarar e instanciar un DataAdapter.
            SqlDataAdapter daFicha = new SqlDataAdapter();
            //Configurar la propiedad (SelectCommand).
            daFicha.SelectCommand = spConsulta;
```

```
            daFicha.SelectCommand.CommandType = CommandType.StoredProcedure;


            daFicha.SelectCommand.Parameters.Add("@Descripcion",
SqlDbType.NVarChar, 100);
            daFicha.SelectCommand.Parameters["@Descripcion"].Value = des;


            //Declarar e instanciar un DataSet.
            DataSet dsDatos = new DataSet();


            //Recuperar los datos.
            daFicha.Fill(dsDatos, "Productos");
            return dsDatos;
        }
        catch (SqlException error)
        {
            throw error;
        }
        catch (Exception error)
        {
            throw error;
        }
        finally
```

```
        {
            conexion.Close();
            conexion.Dispose();
        }
    }
```

// **Select(sin parametros),- Este metodo obtiene todos los datos de los productos sin ninguna excepción.**

```
    public DataSet Select()
    {
        SqlConnection conexion = new SqlConnection();

        try
        {
            //Configurar el Connection.
            conexion.ConnectionString = claseConexion.Conexion;

            SqlDataAdapter daCategoria = new SqlDataAdapter("SELECT * FROM vistaVenta", conexion);
            SqlDataAdapter daDetalle = new SqlDataAdapter("SELECT * FROM vistaDetalleVenta", conexion);
```

```
            DataSet dsVenta = new DataSet();


            daCategoria.Fill(dsVenta, "vistaVentas");
            daDetalle.Fill(dsVenta, "vistaDetalleVenta");


            return dsVenta;
        }
        catch (SqlException error)
        {
            throw error;
        }
        catch (Exception error)
        {
            throw error;
        }
        finally
        {
            conexion.Dispose();
        }
    }
}
```

}

6.6.3.2 Script de las Tablas

TABLA CATEGORIAS

```
USE [Ferreteria]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Categorias](
                                                    [idCategoria]
[int] IDENTITY(1,1) NOT NULL,
                                                    [Descripcion]
[nvarchar](100) COLLATE Modern_Spanish_CI_AS NULL,
 CONSTRAINT [PK_Categorias] PRIMARY KEY CLUSTERED
(
                                                    [idCategoria]
ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

TABLA PRODUCTOS

```
USE [Ferreteria]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Productos](
                                                    [idProducto] [int]
IDENTITY(1,1) NOT NULL,
                                                    [idCategoria]
[int] NULL,
                                                    [Descripcion]
[nvarchar](150) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Unidad]
[nvarchar](10) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Existencia] [int]
NULL,
                                                    [PrecioCompra]
[float] NULL,
                                                    [PrecioVenta]
[float] NULL,
 CONSTRAINT [PK_Productos] PRIMARY KEY CLUSTERED
```

(

[idProducto]

ASC

)WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Productos] WITH CHECK ADD CONSTRAINT [FK_Productos_Categorias] FOREIGN KEY([idCategoria])

REFERENCES [dbo].[Categorias] ([idCategoria])

GO

ALTER TABLE [dbo].[Productos] CHECK CONSTRAINT [FK_Productos_Categorias]

TABLA CLIENTES

USE [Ferreteria]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Clientes](

```
                                                    [CI]    [nchar](10)
COLLATE Modern_Spanish_CI_AS NOT NULL,
                                                    [Nombres]
[nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Apellidos]
[nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Direccion]
[nvarchar](100) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Telefono]
[nvarchar](12) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Tipo]
[nvarchar](12) COLLATE Modern_Spanish_CI_AS NULL,
 CONSTRAINT [PK_Clientes] PRIMARY KEY CLUSTERED
(
                                                    [CI] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

TABLA PROVEEDORES

```
USE [Ferreteria]
GO
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Proveedores](

                                                    [CI]    [nchar](10)
COLLATE Modern_Spanish_CI_AS NOT NULL,

                                                    [Nombres]
[nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [Apellidos]
[nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [Direccion]
[nvarchar](100) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [Telefono]
[nvarchar](10) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [Empresa]
[nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL,

 CONSTRAINT [PK_Proveedores] PRIMARY KEY CLUSTERED

(

                                                    [CI] ASC

)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]
```

TABLA CHEQUES

```
USE [Ferreteria]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Cheques](
                                                    [IdCheque]    [int]
IDENTITY(1,1) NOT NULL,
                                                    [NumeroFactura]
[nchar](10) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Nombre]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Banco]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [NumeroCuenta]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [NumeroCheque]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Fecha]
[datetime] NULL,
```

```
                                                    [Valor]      [float]
NULL,
 CONSTRAINT [PK_Cheques] PRIMARY KEY CLUSTERED
(
                                                    [IdCheque] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
```

TABLA COMPRAS

```
USE [Ferreteria]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Compras](
                                                    [idCompra]   [int]
IDENTITY(1,1) NOT NULL,
```

```
                                                    [NoFactura]
[nchar](10) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [CI_Proveedor]
[nchar](10) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [FechaEmision]
[datetime] NULL,
                                                    [FechaVencimien
to] [datetime] NULL,
                                                    [Ciudad]
[nvarchar](20) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Lugar]
[nvarchar](20) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Vendedor]
[nvarchar](30) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Subtotal]
[decimal](18, 2) NULL,
                                                    [Iva]
[decimal](18, 2) NULL,
                                                    [Total]
[decimal](18, 2) NULL,
                                                    [Facturado]   [bit]
NULL,
 CONSTRAINT [PK_Compras] PRIMARY KEY CLUSTERED
(
                                                    [idCompra] ASC
```

)WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Compras] WITH CHECK ADD CONSTRAINT [FK_Compras_Proveedores] FOREIGN KEY([CI_Proveedor])

REFERENCES [dbo].[Proveedores] ([CI])

GO

ALTER TABLE [dbo].[Compras] CHECK CONSTRAINT [FK_Compras_Proveedores]

TABLA DETALLE COMPRA

USE [Ferreteria]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DetalleCompra](

[idDetalleCompr

a] [int] IDENTITY(1,1) NOT NULL,

```
                                                    [idCompra]    [int]
NULL,
                                                    [idProducto]  [int]
NULL,
                                                    [Cantidad]    [int]
NULL,
                                                    [PrecioUnitario]
[float] NULL,
                                                    [Descuento]
[float] NULL,
                                                    [Total]       [float]
NULL,
 CONSTRAINT [PK_DetalleCompra] PRIMARY KEY CLUSTERED
(
                                                    [idDetalleCompr
a] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]


GO
ALTER TABLE [dbo].[DetalleCompra]  WITH CHECK ADD  CONSTRAINT
[FK_DetalleCompra_Compras] FOREIGN KEY([idCompra])
REFERENCES [dbo].[Compras] ([idCompra])
GO
```

ALTER TABLE [dbo].[DetalleCompra] CHECK CONSTRAINT [FK_DetalleCompra_Compras]

GO

ALTER TABLE [dbo].[DetalleCompra] WITH CHECK ADD CONSTRAINT [FK_DetalleCompra_Productos] FOREIGN KEY([idProducto])

REFERENCES [dbo].[Productos] ([idProducto])

GO

ALTER TABLE [dbo].[DetalleCompra] CHECK CONSTRAINT [FK_DetalleCompra_Productos]

TABLA VENTAS

USE [Ferreteria]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Ventas](

[idVenta]

[nchar](10) COLLATE Modern_Spanish_CI_AS NOT NULL,

[CI_Cliente]

[nchar](10) COLLATE Modern_Spanish_CI_AS NULL,

```
                                                    [FechaEmision]
[datetime] NULL,
                                                    [FechaVencimien
to] [datetime] NULL,
                                                    [Ciudad]
[nvarchar](20) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Lugar]
[nvarchar](20) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Vendedor]
[nvarchar](30) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Subtotal]
[decimal](18, 2) NULL,
                                                    [Iva]
[decimal](18, 2) NULL,
                                                    [Total]
[decimal](18, 2) NULL,
                                                    [Facturado]   [bit]
NULL,
 CONSTRAINT [PK_Ventas] PRIMARY KEY CLUSTERED
(
                                                    [idVenta] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'FormaPago' , @level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Ventas',
@level2type=N'COLUMN',@level2name=N'Lugar'

GO

ALTER TABLE [dbo].[Ventas] WITH CHECK ADD CONSTRAINT [FK_Ventas_Clientes] FOREIGN KEY([CI_Cliente])

REFERENCES [dbo].[Clientes] ([CI])

GO

ALTER TABLE [dbo].[Ventas] CHECK CONSTRAINT [FK_Ventas_Clientes]

TABLA DETALLEVENTA

USE [Ferreteria]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DetalleVenta](

[idDetalleVenta]

[int] IDENTITY(1,1) NOT NULL,

```
                                                    [idVenta]
[nchar](10) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [idProducto] [int]
NULL,
                                                    [Cantidad]   [int]
NULL,
                                                    [PrecioUnitario]
[float] NULL,
                                                    [Descuento]
[float] NULL,
                                                    [Total]      [float]
NULL,
 CONSTRAINT [PK_DetalleVenta] PRIMARY KEY CLUSTERED
(
                                                    [idDetalleVenta]
ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]


GO
ALTER TABLE [dbo].[DetalleVenta]  WITH CHECK ADD  CONSTRAINT
[FK_DetalleVenta_Productos] FOREIGN KEY([idProducto])
REFERENCES [dbo].[Productos] ([idProducto])
GO
```

ALTER TABLE [dbo].[DetalleVenta] CHECK CONSTRAINT [FK_DetalleVenta_Productos]

GO

ALTER TABLE [dbo].[DetalleVenta] WITH CHECK ADD CONSTRAINT [FK_DetalleVenta_Ventas] FOREIGN KEY([idVenta])

REFERENCES [dbo].[Ventas] ([idVenta])

GO

ALTER TABLE [dbo].[DetalleVenta] CHECK CONSTRAINT [FK_DetalleVenta_Ventas]

TABLA RETENCIONCOMPRA

USE [Ferreteria]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_PADDING ON

GO

CREATE TABLE [dbo].[RetencionCompra](

[NoRetencionCompra] [nchar](10) COLLATE Modern_Spanish_CI_AS NOT NULL,

[Señor] [varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

[RUC]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

[Direccion]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

[FechaEmision]
[datetime] NULL,

[TipoComproban
te] [varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

[NoComprobante
] [varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

[Total]
[decimal](18, 2) NULL,

CONSTRAINT [PK_RetencionCompra] PRIMARY KEY CLUSTERED

(

[NoRetencionCo
mpra] ASC

)WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI_PADDING OFF

TABLA RETENCIONCOMPRADETALLE

USE [Ferreteria]

```
GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_PADDING ON

GO

CREATE TABLE [dbo].[RetencionCompraDetalle](

                                                    [IdDetalleRetenci
on] [int] IDENTITY(1,1) NOT NULL,

                                                    [NoRetencionCo
mpra] [nchar](10) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [EjercicioFiscal]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [BaseImponible]
[float] NULL,

                                                    [CompPago]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [Codigo]
[nchar](10) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [PorcentajeReten
cion] [varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                    [ValorRetenido]
[float] NULL,
```

```
 CONSTRAINT [PK_RetencionCompraDetalle] PRIMARY KEY CLUSTERED
(
                                                    [IdDetalleRetenci
on] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[RetencionCompraDetalle]  WITH CHECK ADD  CONSTRAINT [FK_RetencionCompraDetalle_RetencionCompra] FOREIGN KEY([NoRetencionCompra])
REFERENCES [dbo].[RetencionCompra] ([NoRetencionCompra])
GO
ALTER TABLE [dbo].[RetencionCompraDetalle] CHECK CONSTRAINT [FK_RetencionCompraDetalle_RetencionCompra]
```

TABLA RETENCIONVENTA

```
USE [Ferreteria]
GO
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON

GO

SET ANSI_PADDING ON

GO

CREATE TABLE [dbo].[RetencionVenta](

                                                                [NoRetencionVe
nta] [nchar](10) COLLATE Modern_Spanish_CI_AS NOT NULL,

                                                                [Señor]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                                [RUC]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                                [Direccion]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                                [FechaEmision]
[datetime] NULL,

                                                                [TipoComproban
te] [varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                                [NoComprobante
] [varchar](50) COLLATE Modern_Spanish_CI_AS NULL,

                                                                [Total]
[decimal](18, 2) NULL,

 CONSTRAINT [PK_RetencionVenta] PRIMARY KEY CLUSTERED

(
```

[NoRetencionVe

nta] ASC

)WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI_PADDING OFF

TABLA RETENCIONVENTADETALLE

USE [Ferreteria]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_PADDING ON

GO

CREATE TABLE [dbo].[RetencionVentaDetalle](

[IdDetalleRetenci

on] [int] IDENTITY(1,1) NOT NULL,

[NoRetencionVe

nta] [nchar](10) COLLATE Modern_Spanish_CI_AS NULL,

```
                                                    [EjercicioFiscal]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [BaseImponible]
[float] NULL,
                                                    [CompPago]
[varchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [Codigo]
[nchar](10) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [PorcentajeReten
cion] [varchar](50) COLLATE Modern_Spanish_CI_AS NULL,
                                                    [ValorRetenido]
[float] NULL,
 CONSTRAINT [PK_RetencionVentaDetalle] PRIMARY KEY CLUSTERED
(
                                                    [IdDetalleRetenci
on] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[RetencionVentaDetalle]  WITH CHECK ADD  CONSTRAINT
[FK_RetencionVentaDetalle_RetencionVenta] FOREIGN KEY([NoRetencionVenta])
```

REFERENCES [dbo].[RetencionVenta] ([NoRetencionVenta])

GO

ALTER TABLE [dbo].[RetencionVentaDetalle] CHECK CONSTRAINT [FK_RetencionVentaDetalle_RetencionVenta]

TABLA AUDITORIAPRODUCTOS

USE [Ferreteria]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AuditoriaProductos](

[idProducto] [nchar](10) COLLATE Modern_Spanish_CI_AS NULL,

[Descripcion] [nvarchar](100) COLLATE Modern_Spanish_CI_AS NULL,

[Usuario] [nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL,

[Fecha] [datetime] NULL,

[Accion] [nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL

) ON [PRIMARY]

6.6.4 Pruebas

6.6.4.1 Pruebas de Caja Blanca

Las pruebas de caja blanca se realizaron a nivel de base de datos y de los módulos de clase.

Base de datos.- En la base de datos se vio la necesidad de implementar una función especial, la misma que calculará el precio de venta, con un porcentaje de incremento respecto al precio de compra, esto con el objetivo de agilizar el proceso en el momento de ingresar los productos, ya que en instancias iníciales se lo hacía directamente en el ingreso, y el usuario podría haber cometido algún error en el cálculo del mismo acarreando graves problemas por tal situación. También se debió implementar funciones en las tablas de compra y venta, para controlar las fechas de ingreso de la compra y vencimiento, ya que el usuario podría ingresar cualquier fecha, y al instante de procesarle podrían ser incoherentes como por ejemplo que la fecha de vencimiento sea menor a la fecha de de compra, por lo cual se tuvo algunos errores al momento de realizar las consultas y reportes por fecha, por tal motivo se lo implemento para evitar estas situaciones. En estas mismas tablas se controla que si el usuario no ingresaba ninguna fecha se guardaría con la fecha del sistema, ya que algunas veces el usuario se olvidaba de ingresar mencionadas fechas que son importantes para las consultas y reportes por fechas que maneja la empresa.

Clases.- En cuanto a las pruebas en las clases, se tuvo de implementar métodos para búsquedas individuales por código utilizando sentencias SQL, en lugar de búsquedas secuenciales registro por registro, ganando con esto mayor velocidad al momento de procesar las consultas. Esto se tuvo que realizar en todas las clases de acceso a datos. En la clase de compras y ventas se creó los métodos de actualización de existencia, esto con el fin de tener una tabla de productos actualizada, ya que cuando se realizaba una compra o venta, los productos mantenían la misma existencia, al mismo tiempo se implementó los métodos

para revertir dichas actualizaciones que se pudieron cometer errores en la digitación de los detalles de las facturas lo cual crearía una existencia equívoca, y por tal problemas a la empresa, con el método revertir se logró resolver dicho inconveniente y funcionamiento adecuado.

6.6.4.2 Pruebas de Caja Negra

La prueba verifica que el ítem que se está probando, cuando se dan las entradas apropiadas produce los resultados esperados.

Esta prueba se realizo sobre la interfaz final, por lo tanto estas pruebas son completamente indiferentes del comportamiento interno y la estructura de las capas lógicas del sistema.

Probamos que los cálculos de las facturas trabajen de la manera correcta, los datos de ingreso sean verdaderos por ejemplo la cedula valida, las fechas.

Estas pruebas demostraron que las funciones del sistema son completamente operativas y factibles, registros correctos, obtención de la información de forma adecuada, e integridad de la información.

6.6.5 Implantación

Se realizó una implantación directa, verificando la información manual y el sistema entró en funcionamiento inmediatamente.

La empresa contrato dos personas que colaboren en la parte de ingresos de datos, el proceso duro 2 semanas, y una verificación de 1 semana para comprobar los cálculos, y procesos que correspondan con lo que anteriormente la empresa trabajaba en forma manual.

6.7 Conclusiones y Recomendaciones

6.7.1 Conclusiones

- Para el desarrollo del sistema se analizó los procedimientos de compras, proveedores e inventarios de la empresa, creando procesos que permitan un manejo eficiente de la información al plantear una propuesta informática de solución utilizando base de datos relacional y tecnología, para el control de productos y servicios en la Ferretería Bolivariana.
- Se logro realizar in diseño flexible y de fácil manejo para el usuario, con lo cual la empresa podrá agilizar su trabajo, ahorrando tiempo en las transacciones realizadas y guardando la información de manera segura en la base de datos.
- Las pruebas realizadas sobre el funcionamiento del sistema fueron muy rigurosas para obtener un optimo rendimiento del mismo
- La implantación se realizo de manera directa en el servidor y en el cliente (maquina de bodega) con un motor de base de datos compartida.

6.7.2 Recomendaciones

- Se recomienda a la empresa que realice una adecuada capacitación a las personas que vayan a usar el sistema, en la forma de su uso, esto con el fin de evitar inconvenientes de pérdida de información o ingreso erróneo de la misma.

- También se recomienda que la empresa implemente equipo informático de última tecnología en el servidor de base de datos, ya que con esto se beneficiaría los terminales conectados a ella, con mayor velocidad y rendimiento al momento de realizar transacciones con los datos de la base, acarreando una mejor atención y satisfacción al cliente.

- Realizar Backup de la base de datos una vez por mes, por lo menos esto con el fin de evitar pérdidas sustanciales de información, ya que muchas veces los equipos informáticos suelen ser vulnerables a ciertos daños físicos o lógicos, los mismos que traerían graves problemas a la empresa si la base de datos fuera corrompida, con esto no se perderá dicha información.

6.8 Bibliografía

LIBROS

Fundamentación Teórica de Base de Datos Armando Sánchez

GOMEZ, Ángel Lucas, 2000, Diseño Y Gestión De Sistemas De Bases De Datos, Editorial Paraninfo Madrid, España

DATE, CJ, 1999, Sistemas De Bases De Datos, Quinta Edición EUA

Análisis y Diseño de Sistemas de Información de Kendall y Kendall

6.9 Glosario de Términos

- **Aleatoriedad**.- Dependiente de algún proceso o suceso fortuito.
- **Atributo**.- Cada una de las cualidades o propiedades de un sistema o archivo
- **Cifrado de datos**.- Signo con que se representa los datos para ser procesados.
- **Clase:** definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.
- **Compilación**.- Obra que reúne informaciones, preceptos o doctrinas aparecidas antes por separado o en otras obras
- **Componentes de un objeto**: atributos, identidad, relaciones y métodos.

- **Estado interno**: es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.
- **Evento**: un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.
- **Exportar**.- trasladar información de un sistema a otro.
- **Filtros**.- Sistema de selección en un proceso según criterios previamente establecidos
- **Folder**.- carpeta para almacenar o clasificar información
- **Herencia:** (por ejemplo, herencia de la clase D a la clase C) Es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables publicas declaradas en C. Los componentes registrados como "privados" (private) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Esto es así para mantener hegemónico el ideal de OOP.
- **Interfaz**.- Conexión física y funcional entre dos aparatos o sistemas independientes.
- **Mensaje**: una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.
- **Método:** Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.
- **Multiproceso**.- Técnica para la ejecución simultánea de dos o más procesos en una misma computadora.

- **Objeto:** entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos) los mismos que consecuentemente reaccionan a eventos. Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.
- **Propiedad o atributo**: contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.
- **Representación de un objeto**: un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.
- **Vínculos**.- Enlace de información entre diferentes páginas o archivos