UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS

Sistema Cliente/Servidor aplicado a punto de ventas a través de Sockets para la Ferretería San Agustín.

Autor: Francisco Eduardo Espinel Jerez

Director: Ing. Clay Aldás

Asesor: Ing. Hernando Buenaño

Tesis de Grado, previa a la Obtención del Titulo de Ingeniero en Sistemas

Ambato - Ecuador

Mayo/2005

Agradecimiento

A los señores docentes de la Universidad Técnica de Ambato, Facultad de Ingeniería en Sistemas, que con nobleza y entusiasmo supieron compartir sus conocimientos de manera objetiva y desinteresada siempre dispuestos a satisfacer todas las inquietudes.

A todas y cada una de las personas que durante estos años de carrera universitaria supieron aportar con su grano de arena para la consecución de una formación integral.

A la Empresa Ferretería San Agustín, en la persona del Sr. Marcelo Jerez Garzón, Gerente Propietario, por todo el apoyo brindado para la realización del presente Proyecto.

Dedicatoria

El presente trabajo va dedicado a mis padres quienes a través de su amor supieron encaminar mi vida por el sendero del bien, me brindaron apoyo, confianza para salir siempre adelante y nunca desmayar. A Dios por ser la fuente que ilumina toda mi existencia.

Declaración, Autenticidad y Responsabilidad

Yo, Francisco Eduardo Espinel Jerez declaro que:

La investigaciones enmarcado en el desarrollo de la presente Tesis de Grado, es absolutamente original, autentica y personal. En tal virtud declaro que el contenido, efectos legales y académicos que se desprenden del trabajo son y serán de mi exclusiva responsabilidad legal y académica.

INTRODUCCIÓN

El mundo de la Informática evoluciona muy rápidamente. Cada vez son más las áreas donde la informática juega un papel muy importante e incluso fundamental.

En el proceso de controlar máquinas, las empresas se encuentran con la necesidad de interconectar distintos controladores entre si y a estos con sistemas de información como bases de datos de compras, ventas, inventarios, etc. La interconexión de controladores y sistemas informáticos se conoce como automatización. La automatización tiende hacia la manufactura integrada por computadora la cual da un gran control a la gerencia de la empresa sobre todos los aspectos de producción y permite una satisfacción más flexible de las necesidades del cliente.

En la actualidad toda empresa competitiva trata de automatizar sus departamentos para tener un mejor control de los datos, realizar análisis y orientarse en las formas que puede hacer a su empresa día a día más competitiva y productiva.

A través del presente proyecto se ofrecerá al personal de la Ferretería San Agustín una herramienta clara, comprensible y de fácil uso que agilitara las diferentes actividades comerciales de la empresa, **EL SISTEMA CLIENTE / SERVIDOR APLICADO A PUNTO DE VENTAS A TRAVÉS DE SOCKETS**, mejorara el sistema de información existente, minimizando el tiempo de cada actividad que interviene en los procesos y aprovechando de mejor manera los recursos humanos como materiales; dando una mejor atención al cliente.

Este proyecto se encuentra distribuido en cuatro capítulos, donde el Capitulo uno hace referencia a los antecedentes de la empresa, la forma de trabajo y su desarrollo hasta la actualidad.

En el Capitulo dos se hace referencia al Marco Teórico investigado para el desarrollo del Sistema.

En el Capitulo tres se realiza un análisis y diseño del sistema a ser realizado.

En el Capitulo cuatro se presenta un diseño de entradas y salidas del sistema.

Culminando con las conclusiones y recomendaciones a aplicarse luego del análisis, diseño y realización del sistema propuesto.

Cabe señalar que el sistema propuesto se desarrollara de tal manera que signifique un verdadero aporte para la empresa ratificando la importancia del mismo, ya que su formulación, planificación y consecución constituyen el resultado de un proceso metodológico que integra la teoría con la práctica.

INDICE

CAPITULO III

INDICE DE FIGURAS

1.1. Antecedentes

Ferretería San Agustín, la cual se encuentra en el mercado por más de 20 años sirviendo a la ciudadanía Latacungueña, ha ido creciendo año tras año, convirtiéndose en una empresa líder en el mercado de la ciudad de Latacunga, por su gran variedad de artículos, atención al cliente, servicio de transporte y asesoramiento técnico, siempre trata que sea un ambiente limpio, ordenado y bien iluminado.

Ferretería San Agustín es representante de tubería PVC para riego, Hierro, Cemento Selva Alegre, pone a disposición de su clientela: Tubería y Accesorios PVC, HG, Roscable, Válvulas HF-RW, Collarines, Medidores, Bombas para agua, Pinturas, Mallas, Cerámica, Sanitarios, Etc.

El crecimiento de la empresa ha conllevado a la automatización y tecnificación de los departamentos de la empresa para un mejor servicio al cliente, así como un mayor control del inventario en cada una de las bodegas que son parte de la empresa.

Por el volumen, espacio y tipo de productos que se despachan diariamente, la ferretería esta conformada por tres bodegas y su almacén principal que se encuentra en el centro de la ciudad de Latacunga, cada una de las bodegas esta ubicada en diferentes sectores de la ciudad, cada bodega cuenta con un bodeguero y un despachador, el almacén principal cuenta con los departamentos de Secretaria, Ventas y Administración.

La realización de ventas, facturación, entrega de proformas se lo realiza en forma manual, presentando varios inconvenientes como:

- El tiempo de atención al cliente, ya que la factura se la realiza manualmente.

- Verificación de mercadería en la bodega, pues el vendedor no sabe con exactitud la cantidad de mercadería disponible.

- Anulación de facturas, por errores producidos al llenar la factura.

- Dificultad en la verificación de ítems y precios, debido a que se cuenta con un listado de productos el cual a veces no esta actualizado.

Al producirse venta, se realiza la entregada de la factura o nota de venta al cliente, este tiene que acercarse a caja a realizar la cancelación de la misma.

Tomando en cuenta la forma de trabajo, el despacho de mercadería, el control que se debe dar al inventario, la forma en que se realiza los cobros y necesidades de este tipo de negocios se propone la creación del Sistema Cliente/Servidor aplicado a puntos de ventas a través de Sockets, el cual tendrá las siguientes características principales:

- Controlar todos los vendedores, clientes y proveedores que se desee, independientemente uno del otro.

- Actualizar el inventario en línea, permitiendo un control exacto de las existencias en cada una de las bodegas.

- Manejar varios precios de venta por producto.

- Permitir la selección del producto por código, nombre.

- Recibir mercadería y realizar el cálculo del precio de venta basado en márgenes que se desee para el producto.

- Facturar las ventas.

- Proformar mercadería para luego ser vendida.

- Presentación y almacenamiento de la información.

A continuación se muestra una esquema representativo de cómo estarán conectados los puntos de venta y en si todos los nodos que conforman la red, esto se ilustra en la Figura 1.1.

Figura 1.1. Esquema representativo del Sistema Cliente/Servidor aplicado a puntos de venta a través de Sockets.

El desarrollo del Sistema permitirá recoger, gestionar, controlar y difundir la información de toda una empresa u organización.

El Sistema Cliente / Servidor aplicado a punto de ventas a través de Sockets, tiene la capacidad de controlar las Ventas, el Inventario, Compras, Clientes – Cuentas x Cobrar, Proveedores – Cuentas x Pagar, Caja, Bancos, Facturación, y en general, todo lo relacionado a la operación del Negocio, se puede usarlo en una computadora o en red, no necesita de una persona experimentada en el uso de Sistemas de información ya que su manipulación es simple, practica, sencilla, capaz de que cualquier persona pueda hacer uso de este Sistema sin la necesidad de un manual para el usuario, el Sistema dispone de una base de datos segura, permitiendo a los administradores disponer de la información en tiempo real en el momento que se lo requiera, lo cual permitirá tomar decisiones mas acertadas y proyectar un futuro exitoso para la Empresa.

A parte de los aspectos que se han tomado en cuenta en este tipo de negocios se ha sumado otros aspectos como:

- El software existente en el mercado es muy sencillo, costoso y generalizado debido a que se necesita conocer el movimiento de la empresa, así como también se busca una optimización de recursos.

- Adentrarse en los conceptos de Sistema Cliente/Servidor, Sockets y demostrar la factibilidad de la implementación del Sistema Cliente / Servidor aplicado a punto de ventas a través de Sockets.

1.2. Objetivos

1.2.1. Objetivo General

- Realizar un Sistema Cliente/Servidor aplicado a puntos de ventas a través de Socktes.

1.2.2. Objetivos Específicos

- Controlar la mercadería existente de forma rápida, eficiente y real.
- Reducir costos y tiempo empleado en el control del inventario.
- Realizar la facturación por medio de puntos de venta.

1.3. Estudio del Sistema Actual

Ferretería San Agustín actualmente viene llevando el control de inventario, ventas, facturación; con el Sistema Contable Mónica 5.0, este Sistema a presentado el gran inconveniente de no ser apto para el manejo en red de computadores lo cual hoy en día es indispensable ya que la información debe estar distribuida en todos los equipos de la red, además la necesidad de la empresa es disponer de puntos de venta independientes para dar una mejor atención al cliente, servirle de manera eficaz, donde cada punto de venta atenderá exclusivamente a un cliente. Para lo cual se necesita que la

información deba estar actualizada al momento y ser transparente a los demás puntos de venta teniendo de esta manera un inventario en línea lo cual es un aspecto fundamental al que se da la mayor prioridad posible.

También se ha presentado ciertas inconformidades como: la presentación que a veces tiende a ocasionar errores humanos, el no ser apto para llevar el control de inventario con bodegas.

1.4. Análisis Estructurado

1.4.1. Tabla de Eventos

Tabla de Eventos

Sistema Cliente/Servidor aplicado a puntos de venta a través de Sockets.

Tarea N.-	Descripción del evento	Documento de entrada	Documento de salida	Origen	Destino
1	Ingreso de mercadería a bodega	Factura		Bodega de la Empresa	Departamento de Contabilidad
2	Ajuste de mercadería en bodega	Factura	Nota de ingreso		Departamento de Administración
3	Venta de mercadería a clientes		Factura	Vendedor	Cliente
4	Cuentas por cobrar		Factura		Cliente
5	Compra de mercadería a proveedores	Pedido		Departamento de Administración	
6	Validación de proformas	Proforma		Cliente	Departamento de Administración
7	Facturar proformas		Factura		Departamento de Administración

Ferretería San Agustín en su almacén central dispone de cinco computadores, donde solo dos computadores tienen instalado el Sistema Contable Mónica 5.0, haciendo uso del Sistema en forma separada el personal del departamento de Secretaria.

Claramente se ve un desperdicio de recursos tecnológicos, además el resto de computadores se los usa para la realización de otras actividades, no explotando así dicho recursos a su máximo.

En base a todos estos inconvenientes y percances que presenta el Sistema Contable Mónica 5.0, se realizara el desarrollo del Sistema Cliente/Servidor aplicado a puntos de venta a través de Sockets.

1.5. Propuesta del Sistema

1.5.1. Descripción del Sistema Propuesto

Ya que nos encontramos en un mundo globalizado, con un gran avance tecnológico, donde se ha demostrado que la información es el pilar fundamental para el desarrollo y crecimiento de un negocio se ha realizado la creación del Sistema Cliente/Servidor aplicado a puntos de venta a través de Sockets.

Utilizando tecnología de punta en el diseño y la programación, se ha generado una herramienta que responde exactamente a las expectativas y metodologías administrativas de las empresas exitosas.

Ítems Destacables

Tecnología

- Arquitectura abierta: Para explotar toda la información vía ODBC/SQL y herramientas compatibles.

- Cliente-Servidor: Asegura consistencia en los datos, tolerancia a fallos y velocidad de proceso.

- Windows 2000 Advanced Server: Ofrece mayor escalabilidad y disponibilidad del Sistema. Esta versión de Windows 2000 está diseñada para los servidores que se emplean en una red empresarial de gran tamaño y para tareas que hacen un uso intensivo de base de datos.

- **Sistema Cliente/Servidor aplicado a puntos de venta a través de Sockets es Cliente/Servidor:** La administración de las transacciones y las reglas del negocio son procesadas en el servidor mismo. Esta característica lo hace único en su categoría, diferenciándolo completamente de aquellas aplicaciones que utilizan la base de datos al sólo efecto de almacenar información.

- **Conexión:** La comunicación entre caja y los puntos de venta se lo realiza a través de sockets.

- **Escalabilidad:** Su diseño original y su base resistente permiten incorporar nuevos usuarios, sin resentir la eficiente performance del Sistema.

- **Facturación:** Cada punto de venta factura y actualiza el inventario.

Niveles de seguridad

La información confidencial o reservada así como también determinadas funciones pueden ser acotadas para determinados usuarios.

Funcionalidad

Ventas

- Toma y control de pedidos.

- Control de existencias.
- Facturación.
- Listas de precios estándar, oferta y mínima.

Compras

- Administración de requerimientos.
- Proveedores habituales.
- Generación, emisión y seguimiento de órdenes de compra.
- Ingreso y control de facturas de proveedores.
- Control de recepción de mercaderías.

Cuentas a Cobrar y Pagar

- Presentación de las cuentas pendientes por cobrar y pagar.
- Cálculo y administración de las cuentas.

2.1. Marco Teórico

2.2. Sistema Cliente/Servidor

El modelo cliente/servidor, es el modelo de ejecución que siguen todas las aplicaciones de red. Un servidor es un proceso que se está ejecutando en un nodo de la red, y su función es gestionar el acceso a un determinado recurso. Un cliente es un proceso que se ejecuta en el mismo nodo, o en uno diferente, y que realiza peticiones al servidor. Las peticiones están originadas por la necesidad de acceder al recurso que gestiona el servidor.

2.2.1. Características del Modelo Cliente/Servidor

En el modelo CLIENTE/SERVIDOR podemos encontrar las siguientes características:

1. El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

2. Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.

3. Un servidor da servicio a múltiples clientes en forma concurrente.

4. Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

5. La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

    - Un Sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo Sistema a las estaciones Clientes**.** Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo Sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

    - También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

    - Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

6. Además se constituye como el nexo de unión mas adecuado para reconciliar los Sistemas de información basados en mainframes o mini computadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.

7. Designa un modelo de construcción de Sistemas informáticos de carácter distribuido.

8. Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del Sistema central de información de la organización, al tiempo que puede acceder a los recursos de este host central y otros Sistemas de la organización ponen a su servicio.

2.2.2. Ventajas y Desventajas del Modelo Cliente/Servidor

Uno de los aspectos que más ha promovido el uso de Sistemas Cliente/Servidor, es la existencia de plataformas de hardware cada vez más baratas. Esta constituye a su vez una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en Sistemas grandes. Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

- El esquema Cliente/Servidor facilita la integración entre Sistemas diferentes y comparte información permitiendo, por ejemplo que las máquinas ya existentes puedan ser utilizadas pero utilizando interfaces mas amigables al usuario. De esta manera, podemos integrar PCs con Sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo Sistema operacional.

- Al favorecer el uso de interfaces gráficas interactivas, los Sistemas construidos bajo este esquema tienen mayor interacción más intuitiva con el usuario. El uso de interfaces gráficas para el usuario, el esquema

Cliente/Servidor presenta la ventaja, con respecto a uno centralizado, de que no es siempre necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.

- Una ventaja adicional del uso del esquema Cliente/Servidor es que es más rápido el mantenimiento y el desarrollo de aplicaciones, pues se pueden emplear las herramientas existentes (por ejemplo los servidores de SQL o las herramientas de más bajo nivel como los sockets).

- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

- El esquema Cliente/Servidor contribuye además, a proporcionar, a los diferentes departamentos de una organización, soluciones locales, pero permitiendo la integración de la información relevante a nivel global.

El esquema Cliente/Servidor tiene algunos inconvenientes que se mencionan a continuación:

- Es importante que los clientes y los servidores utilicen el mismo mecanismo (por ejemplo sockets), lo cual implica que se deben tener mecanismos generales que existan en diferentes plataformas.

- Además, hay que tener estrategias pare el manejo de errores y para mantener la consistencia de los datos. La seguridad de un esquema Cliente/Servidor es otra preocupación importante. Por ejemplo, se deben

hacer verificaciones en el cliente y en el servidor. También se puede recurrir a otras técnicas como el encriptamiento.

- El desempeño es otro de los aspectos que se deben tener en cuenta en el esquema Cliente/Servidor. Problemas de este estilo pueden presentarse por congestión en la red, dificultad de tráfico de datos, etc.

- Un aspecto directamente relacionado con lo anterior es el de cómo distribuir los datos en la red. En el caso de una organización, por ejemplo, éste puede ser hecho por departamentos, geográficamente, o de otras maneras. Hay que tener en cuenta que en algunos casos, por razones de confiabilidad o eficiencia, se pueden tener datos replicados, y que puede haber actualizaciones simultáneas.

2.2.3. Comunicación entre Cliente y Servidor

La comunicación entre cliente y servidor, puede ser o bien orientada a la conexión, o bien sin conexión. En el caso de que la comunicación sea orientada a la conexión, esta se lleva a cabo mediante el establecimiento de circuitos virtuales entre el cliente y el servidor. En este caso, el intercambio de información se realiza con una alta fiabilidad fluyendo la información a través del circuito virtual de una forma secuencial, es decir, tiene un flujo continuo. Esto no ocurre en el caso de que la comunicación sea sin conexión, puesto que aquí el intercambio de información se efectúa mediante el envió de datagramas. La fiabilidad es menor, y al contrario que en el caso anterior, los datagramas no siguen un flujo continuo.

La comunicación sin conexión presenta un aspecto simétrico en la medida de que el iniciador del diálogo puede ser cualquiera de los dos que intervienen.

Esto no ocurre en el caso de la comunicación orientada a la conexión, ya que uno de los dos procesos (en posición de cliente) pregunta al otro (en posición de servidor) si acepta esta comunicación.

Dado que la comunicación que normalmente se utiliza es orientada a la conexión, describiremos a continuación cómo sería el comportamiento tanto del cliente como del servidor con este tipo de servicio, mostrando la secuencia de llamadas de cliente y servidor para un servicio sin conexión al final.

2.2.3.1 El servidor

El servidor está continuamente esperando peticiones de servicio. Cuando se produce una petición, el servidor despierta y atiende al cliente. Cuando el servicio concluye, el servidor vuelve al estado de espera. De acuerdo con la forma de prestar el servicio, podemos considerar dos tipos de servidores:

- *Servidores interactivos:* El servidor no sólo recoge la petición de servicio, sino que él mismo se encarga de atenderla. Esta forma de trabajo presenta un inconveniente; si el servidor es lento en atender a los clientes y hay una demanda de servicio muy elevada, se van a originar unos tiempos de espera muy grandes.

- *Servidores concurrentes.* El servidor recoge cada una de las peticiones de servicio y crea otros procesos para que se encarguen de atenderlas. Este tipo de servidores sólo es aplicable en Sistemas multiproceso, como UNIX. La ventaja que tiene este tipo de servicio es que el servidor puede recoger peticiones a muy alta velocidad, porque está descargado de la tarea de atención al cliente. En las aplicaciones donde los tiempos

de servicio son variables, es recomendable implementar este tipo de servidores.

Su papel es pasivo en el establecimiento de la comunicación, ya que después de haber avisado al Sistema al que pertenece que está preparado para responder a las peticiones de servicio, el servidor se pone a la espera de peticiones de conexión que provengan de clientes. Para esto dispone de un socket de escucha, enlazado al puerto TCP (Protocolo de Control de Transmisión) correspondiente al servicio, sobre el que espera las peticiones de conexión. Cuando llega al Sistema una petición de este tipo, se despierta al proceso servidor y se crea un nuevo socket, que se llama socket de servicio, el cual se conecta al cliente. Entonces el servidor podrá, por una parte delegar el trabajo necesario para la realización del servicio a un nuevo proceso (creado por fork) que utilizará entonces la conexión, y por otra parte volverá al socket de escucha.

Después de la aceptación de la comunicación, el servidor tiene dos posibilidades:

- Hacerse cargo de ella, lo que significa eventualmente que otras conexiones pendientes no serán efectivamente aceptadas hasta que el servicio demandado haya sido satisfecho.

- Bien subtratar la gestión de la conexión y la realización del servicio mediante un proceso hijo.

La secuencia de primitivas para la utilización de sockets que el servidor tiene que usar, y su orden, se muestra en la Figura 2.1.

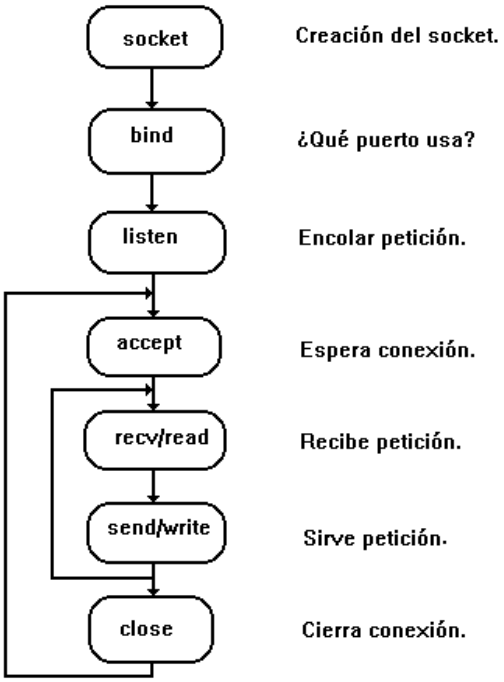


Figura 2.1. Secuencia de llamadas hecha por un servidor.

En el caso de que la comunicación sea sin conexión, la secuencia de llamadas varía sensiblemente, puesto que no es necesario encolar las peticiones que le llegan al servidor, ni esperar a que la conexión se efectúe. El servidor, una vez conectado al puerto correspondiente, se bloquea hasta que recibe alguna petición por parte de un cliente, contestando a este, y retomando la escucha a la espera de nuevas peticiones. La secuencia de llamadas se muestra en la Figura 2.2.

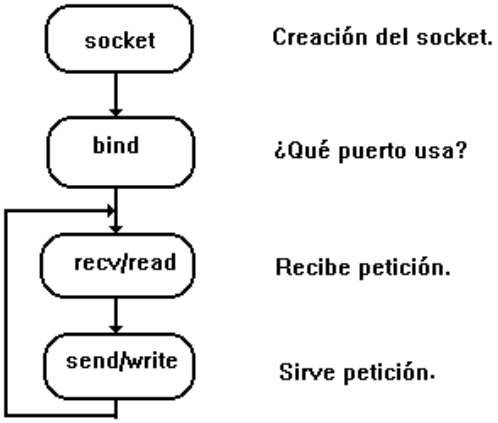


Figura 2.2. Secuencia de llamadas hecha por servidor en una comunicación sin conexión.

2.2.3.2. El cliente

El cliente es la entidad activa en el establecimiento de una conexión, puesto que es el que toma la iniciativa de la demanda de conexión a un servidor. Esta demanda se realiza por medio de la primitiva connect, solicitando el establecimiento de una conexión que será conocida por los dos extremos. Además, el cliente está informado del éxito o del fracaso del establecimiento de la conexión.

Para que un proceso cliente inicie una conexión con un servidor a través de un socket, es necesario realizar una llamada a connect. Así, se crea un circuito virtual entre los dos procesos cuyos extremos son los sockets. La secuencia de primitivas para la utilización de sockets que el servidor tiene que usar, se muestra en la Figura 2.3.

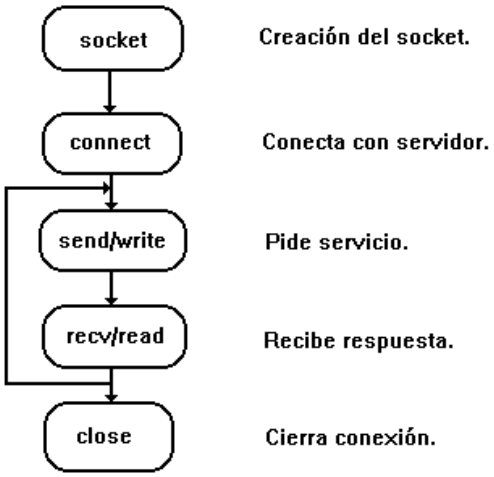


Figura 2.3. Secuencia de llamadas hecha por un cliente.

Pero en el caso de una comunicación sin conexión, el cliente no utiliza la llamada connect para establecer un circuito virtual entre él y el servidor, sino que lo único que hace es conectarse a un puerto por el cual envía peticiones de servicio a un servidor. La secuencia de llamadas, se muestra a continuación Figura 2.4.

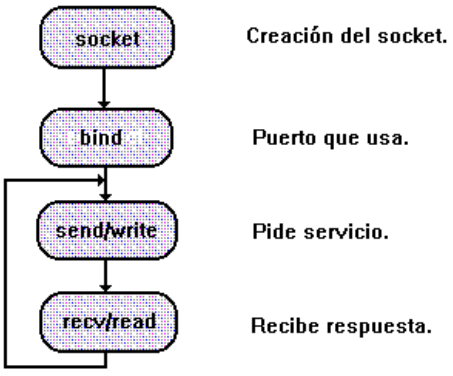


Figura 2.4. Secuencia de llamadas hecha por un cliente en una comunicación sin conexión.

2.2.3.3. Comunicación Cliente/Servidor Orientada a la Conexión.

Una vez establecida la conexión entre un servidor y un cliente a través de dos sockets, los dos procesos pueden intercambiar flujos de información. El corte en diferentes mensajes no está preservado en el socket destino. Esto significa que el resultado de una operación de lectura puede provenir de la información resultado de varias operaciones de escritura. En el caso de los sockets del dominio Internet, una petición de escritura de una cadena de caracteres larga, puede provocar la partición de esta cadena, siendo accesibles los diferentes fragmentos por el socket destino. La única garantía que proporciona el protocolo TCP es que los fragmentos son accesibles en el orden correcto. Esto implica que la sincronización de una recepción y una emisión en una conexión de un mismo número de elementos no está asegurada por este mecanismo, se muestra a continuación Figura 2.5.

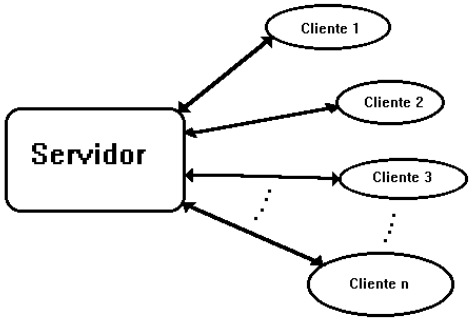


Figura 2.5. Ínter actuación entre clientes y servidor concurrente.

2.3. Sockets

Los sockets son un Sistema de comunicación entre procesos de diferentes máquinas de una red. Más exactamente, un socket es un punto de comunicación por el cual un proceso puede emitir o recibir información.

2.3.1. Tipos de Sockets

Cada tipo de socket va a definir una serie de propiedades en función de las comunicaciones en las cuales está implicado:

a) La fiabilidad de la transmisión. Ningún dato transmitido se pierde.

b) La conservación del orden de los datos. Los datos llegan en el orden en el que han sido emitidos.

c) La no duplicación de datos. Sólo llega a destino un ejemplar de cada dato emitido.

d) La comunicación en modo conectado. Se establece una conexión entre dos puntos antes del principio de la comunicación (es decir, se establece un circuito virtual). A partir de entonces, una emisión desde un extremo está implícitamente destinada al otro extremo conectado.

e) La conservación de los límites de los mensajes. Los límites de los mensajes emitidos se pueden encontrar en el destino.

f) El envío de mensajes (urgentes). Corresponde a la posibilidad de enviar datos fuera del flujo normal, y por consecuencia accesibles inmediatamente (datos fuera de flujo).

Cabe reseñar que un cauce de comunicación normal tiene las cuatro primeras propiedades, pero no las dos últimas.

En cuanto a los tipos de sockets disponibles, se pueden considerar:

SOCK_STREAM: Los sockets de este tipo permiten comunicaciones fiables en modo conectado (propiedades a, b, c y d) y eventualmente autorizan, según el protocolo aplicado los mensajes fuera de flujo (propiedad f). El protocolo subyacente en el dominio Internet es TCP. Se establece un circuito virtual realizando una búsqueda de enlaces libres que unan los dos ordenadores a conectar (parecido a lo que hace la red telefónica conmutada para establecer una conexión entre dos teléfonos). Una vez establecida la conexión, se puede proceder al envío secuencial de los datos, ya que la conexión es permanente. Son streams de bytes full-dúplex (similar a pipes). Un socket stream debe estar en estado conectado antes de que se envíe o reciba en él.

SOCK_DGRAM: Corresponde a los sockets destinados a la comunicación en modo no conectado para el envío de datagramas de tamaño limitado. Las comunicaciones correspondientes tienen la propiedad e. En el dominio Internet, el protocolo subyacente es el UDP. Los datagramas no trabajan con conexiones permanentes. La transmisión por los datagramas es a nivel de paquetes, donde cada paquete puede seguir una ruta distinta, no garantizándose una recepción secuencial de la información.

SOCK_RAW: Permite el acceso a los protocolos de más bajo nivel (por ejemplo, el protocolo IP en el dominio Internet). Su uso está reservado al súper usuario.

SOCK_SEQPACKET: Corresponde a las comunicaciones que poseen las propiedades a, b, c, d y e. Estas comunicaciones se encuentran en el dominio XNS (Protocolo de ordenación).

Siendo los dos primeros tipos de sockets los más utilizados.

2.3.2. Errores de Sockets

La siguiente es una lista de posibles errores devueltos por la llamada a WSAGetLastError, con sus explicaciones. (Los errores están en orden alfabético, ordenados por el nombre macro del error).

WSAEACCES (10013)

Permiso denegado.

Se ha echo un intento de acceder a un socket de una manera prohibida por sus permisos de acceso. Un ejemplo sería usar una dirección broadcast para hacer envíos, sin tener el permiso broadcast ajustado con el uso de setsockopt(SO_BROADCAST).

WSAEADDRINUSE (10048)

Dirección en uso.

Solo un uso de cada dirección de socket (protocolo/dirección IP/puerto) es permitido habitualmente. Este error ocurre cuando una aplicación pretende adjuntar (bind) un socket a una IP/puerto que está ya en uso por un socket existente, o un socket que no ha sido cerrado debidamente, o un socket en proceso de cerrarse.

WSAEADDRNOTAVAIL (10049)

No se puede asignar la dirección requerida.

La dirección requerida no es válida en su contexto. Normalmente ocurre por que un intento de enlazar a una dirección que no es válida en la máquina local.

WSAEAFNOSUPPORT (10047)

La familia de la dirección es incompatible con el protocolo.

Se ha usado una dirección incompatible con el protocolo de la petición. Todos los sockets son creados con una "familia de direcciones" asociada (Ej. AF_INET para protocolos de Internet) y un tipo genérico de protocolo (Ej. SOCK_STREAM). Se devolverá este error si se ha especificado un protocolo incorrecto explícitamente en la llamada al socket, o si una dirección de una familia incorrecta es usada para un socket.

WSAEALREADY (10037)

Operación aún en progreso.

Se ha intentado una operación sobre un socket no bloqueante que ya mantiene una operación en progreso.

WSAECONNABORTED (10053)

Conexión abortada.

EL software ha causado el cierre de la conexión. Una conexión ya establecida, ha sido cerrada por el software del host, posiblemente a causa de haber pasado el tiempo máximo de conexión o a un error en el protocolo.

WSAECONNREFUSED (10061)

Conexión rechazada.

No se puede crear una conexión porque la máquina a la que se ha echo la petición la ha rechazado activamente. Esto se produce normalmente al tratar de conectarse a un servicio que está inactivo (Ej. no hay un servidor funcionando) o porque nuestra conexión viola las normas de seguridad del host (firewall, tcp-wrappers).

WSAECONNRESET (10054)

Conexión reiniciada por el peer.

Una conexión existente ha sido cerrada por el host remoto. (Se ha enviado un paquete con el bit reset). Esto ocurre normalmente cuando una aplicación en el host remoto es apagada de repente o el host es reiniciado. Este error también puede ocurrir si una conexión "keep-alive" es cerrada al detectar un fallo mientras se realizan una o más operaciones.

WSAEDESTADDRREQ (10039)

Se requiere dirección de destino.

Una dirección requerida ha sido omitida en una operación del socket. Por ejemplo, este error será devuelto si se llama a sendto() con la dirección remota ADDR_ANY

WSAEFAULT (10014)

Dirección errónea.

El Sistema ha detectado una dirección de puntero inválido al intentar usar un argumento puntero de una llamada. Este error ocurre si la aplicación pasa un valor de puntero inválido, o si la dirección del buffer es muy pequeña. Por ejemplo, si el tamaño de un argumento que es una estructura sockaddr es menor que sizeof(struct sockaddr).

WSAEHOSTDOWN (10064)

El host está apagado.

Una operación con sockets ha fallado porque el host de destino estaba apagado. La actividad de red en el host local no se inició. Estas condiciones es más fácil que se indiquen con el error WSATIMEDOWN.

WSAEHOSTUNREACH (10065)

No route to host.

Se ha tratado de conectar a un host inalcanzable. (Ver WSAENETUNREACH) Este problema se da, por ejemplo, cuanto la pasarela (gateway) de una red interna o un router que da acceso a esa pasarela están apagados, mientras los ordenadores de esa red tratan de alcanzar un host en la red externa.

WSAEINPROGRESS (10036)

Operación en marcha.

Una operación bloqueante está en ejecución en este momento. Los sockets de Windows permiten una única operación bloqueante por tarea (o hilo de ejecución), y si se realiza otra llamada la función fallará con este error.

WSAEINTR (10004)

Se ha llamado a la función de interrupción.

Una operación bloqueante ha sido interrumpida por una llamada a WSACancelBlockingCall.

WSAEINVAL (10022)

Argumento inválido.

Se ha pasado un argumento inválido (por ejemplo, espeficificar un nivel inválido a la función setsockopt).

WSAEISCONN (10056)

El socket ya está conectado.

Se ha requerido una conexión en un socket ya conectado. Algunas implementaciones también devuelven este error si sendto es llamado en un socket SOCK_DGRAM conectado (Para SOCK_STREAM el par∓aacute:metro en sendto es ignorado, mientras otras implementaciones tratan esto como un suceso legal.

WSAEMFILE (10024)

Demasiados archivos abiertos.

Demasiados sockets. Cada implementación tiene un máximo de punteros a sockets disponibles, tanto globalmente, como por proceso o por tarea.

WSAEMSGSIZE (10040)

Mensaje muy largo.

El mensaje enviado en un socket (datagrama) es más lar que el buffer interno o algún otra limitación de la red, o el buffer usado para recibir un datagrama es menor que el mismo datagrama.

WSAENETDOWN (10050)

La red esta desconectada.

Una operación con sockets encontró una red inutilizable. Esto puede indicar un fallo serio de Sistema de red, la interfase de red o la red local.

WSAENETRESET (10052)

La red abandonó la conexión al reiniciar.

La conexión se ha roto, a causa de un fallo en el progreso de una operación "keep-alive".También puede ser devuelto por setsockopt si se ha intentado ajustar SO_KEEPALIVE en una conexión que ya ha fallado.

WSAENETUNREACH (10051)

La red es inalcanzable.

Se ha intentado una operación con sockets hacia una red inalcanzable. Esto normalmente quiere decir que el software local no conoce ninguna ruta para alcanzar el host remoto.

WSAENOBUFS (10055)

No hay espacio de buffer disponible.

Una operación con sockets no puede ser realizada a causa de que el Sistema no tiene suficiente espacio de buffer o porque la cola esta llena.

WSAENOPROTOOPT (10042)

Mala opción del protocolo.

Una opción o nivel inválidos o no soportados fueron especificados en una llamada a getsockopt o setsockopt.

WSAENOTCONN (10057)

El socket no esta conectado.

Una petición para mandar o recibir datos no esta permitida porque el socket no está conectado (al enviar o recibir un datagrama usando sendto) no se ha especificado una dirección. Cualquier otro tipo de operación podría también devolver este error - por ejemplo, setsockopt, ajustando SO_KEEPALIVE si la conexión se ha reiniciado.

WSAENOTSOCK (10038)

Operación con sockets sobre un no-socket.

Se ha intentado una operación en al que no es un socket. O el parámetro del controlador del socket no referencia un socket válido, o un miembro de un fd_set no es válido.

WSAEOPNOTSUPP (10045)

Operación no soportada.

La operación intentada no es soportada por el tipo de objeto referenciado. Normalmente esto ocurre cuando el descriptor de un socket que no puede

soportar esta operación, por ejemplo, tratar de aceptar una conexión en un socket de datagramas.

WSAEPFNOSUPPORT (10046)

Familia de protocolos no soportada.

La familia de protocolos no ha sido configurada en el Sistema o no existe una implementación para ella. Tiene un significado diferente a WSEAFNOSUPPOT pero es intercambiable en la mayoría de los casos, y todas las funciones de Sckets de Windows que devuelven uno de ellos especifican WSAEAFNOSUPPORT.

WSAEPROCLIM (10067)

Demasiados procesos

Una implementación de sockets de Windows puede tener un límite en el número de aplicaciones que pueden usarla simultáneamente. WSAStartup puede fallar con este error si se alcanza este límite.

WSAEPROTONOSUPPORT (10043)

Protocolo no soportado.

El protocolo pedido no ha sido configurado en el Sistema o no existe una implementación del mismo. Por ejemplo, una llamada con sockets pide un socket SOCK_DGRAM, pero especifica un protocolo de flujo ("stream").

WSAEPROTOTYPE (10041)

Tipo erróneo de protocolo para el socket.

Un protocolo a sido especificado en una llamada a una función con sockets, que no soportan la semántica del tipo de socket solicitado. Por ejemplo, el protocolo "ARPA Internet UDP" no puede especificarse con un socket de tipo SOCK_STREAM.

WSAESHUTDOWN (10058)

No puede enviarse después de cerrar el socket.

Una petición para enviar o recibir datos no fue permitida porque el socket había sido cerrado mediante una llamada anterior. Llamando a shutdown se pide un cierre parcial del socket, lo que es una señal de que enviar o recibir o ambas cosas ha sido descontinuado.

WSAESOCKTNOSUPPORT (10044)

Tipo de socket no soportado.

El soporte para el socket especificado, no existe en esta familia de direcciones. Por ejemplo, el tipo opcional SOCKET_RAW puede ser seleccionado en una llamada con sockets, y la implementación no soportar socket SOCK_RAW en absoluto.

WSAETIMEDOUT (10060)

Timeout de conexión.

Ha fallado un intento de conexión por que la otra parte de la conexión no respondió después de un periodo de tiempo, o la conexión establecida ha fallado, por que el host conectado no ha respondido.

WSATYPE_NOT_FOUND (10109)

Tipo de clase no encontrado.

La clase especificada no fue encontrada.

WSAEWOULDBLOCK (10035)

Recurso no disponible temporalmente.

Este error es devuelto por operaciones en sockets no bloqueantes que no pueden ser completadas inmediatamente, por ejemplo llamar a recv cuando no

hay datos en la lista para ser leídos desde el socket. Es un error no fatal, y la operación puede ser intentada más tarde. Es normal que sea devuelto WSAEWOULDBLOCK como resultado de llamar a connect sobre un soctet no bloqueante del tipo SOCK_STREAM, ya que debe pasar un tiempo para establecer la conexión.

WSAHOST_NOT_FOUND (11001)

No se ha encontrado el host.

No se conoce al host. El nombre no es un alias o nombre de host oficial, o no puede ser hallado en la(s) base(s) de datos que han sido consultadas. Este error también puede ser devuelto por peticiones de protocolos o servicios, y significa que el nombre especificado no puede ser encontrado en la base de datos autorizada.

WSA_INVALID_HANDLE (depende del SO)

El manejador del objeto de eventos es inválido.

Una aplicación intentó usar un objeto de evento, pero el manejador no es válido.

WSA_INVALID_PARAMETER (depende del SO)

Uno o más parámetros son inválidos.

Una aplicación usó un Enchufes de Windows funcionan que directamente traza a la función de Win32. La función de Win32 está indicando un problema con uno o más parámetros.

WSAINVALIDPROCTABLE (depende del SO)

Tabla de procedimientos inválida desde un proveedor de servicios.

Un proveedor de servicios, devolvió una tabla de procedimientos inválida para WS2_32.DLL. (Normalmente es a causa de que uno o más de los punteros a funciones son nulos (=0x00)).

WSAINVALIDPROVIDER (depende del SO)
Número de versión erróneo para el proveedor de servicios.
El proveedor de servicios devolvió un número de versión distinto de 2.0.

2.3.3. La Primitiva Socket

Esta primitiva permite la creación de un socket, es decir, la creación e inicialización de entradas en las diferentes tablas del Sistema de gestión de archivos, que son: tabla de descriptores de procesos, tabla de archivos y estructuras de datos, conteniendo las características del socket. Entre estas características se encuentran:

- El tipo, el dominio y el protocolo.
- El estado del socket (conectado o no, enlazado, en estado de recibir o de emitir).
- La dirección del socket conectado (si hay alguno): al socket se le asocia un buffer de emisión y otro de recepción.
- Punteros a los datos (en emisión y en recepción).
- Un grupo de procesos para la gestión de mecanismos asíncronos.

La forma general de la primitiva que permite crear un socket y obtener un descriptor para utilizarlo es:

```
int socket (dominio, tipo, protocolo)
int dominio; /* AF_INET, AF_UNIX, ... */
int tipo; /* SOCK_DGRAM, SOCK_STREAM, ... */
int protocolo; /* 0: protocolo por defecto */
```

En caso de fallo de la primitiva se devuelve el valor -1 y en error estará codificado el error producido. Si no hay fallo se devuelve un descriptor referenciado al socket, que se utilizará en llamadas posteriores a funciones de la interfaz.

El primer parámetro especifica la familia de sockets que se desea emplear. El segundo parámetro especifica el tipo de socket. El tercer parámetro especifica el protocolo que se va a usar en el socket. Normalmente, cada tipo de socket tiene asociado sólo un protocolo, pero si hubiera más de uno, se especificaría mediante este argumento. Generalmente su valor es 0, en cuyo caso la elección del protocolo se deja en manos del Sistema.

2.3.4. La Primitiva Close

Un socket es suprimido cuando ya no hay ningún proceso que posea un descriptor para accederlo. Esta supresión, la cual es el resultado de al menos una llamada a la primitiva close, implica la liberación de entradas en las diferentes tablas y buffers reservados por el Sistema relacionados con el socket.

2.3.5. La Primitiva Bind

Cuando se crea un socket con la llamada socket, se le asigna una familia de direcciones, pero no una dirección particular. Después de esto, un socket no es accesible más que por los procesos que conocen su descriptor. Sin un mecanismo suplementario de designación, sólo los procesos que hayan heredado tal descriptor en su creación (por la primitiva fork) podrían utilizar un socket. La primitiva bind permite con una sola operación dar un nombre a un socket, es decir, asignar un nombre a un socket sin nombre.

```
int bind (sock, p_direccion, lg)
int sock; /* descriptor de socket */
struct sockaddr *p_direccion; /* puntero de memoria a la dirección*/
int lg; /* longitud de la dirección */
```

Bind hace que el socket, cuyo descriptor es sock, se una a la dirección de socket específica en la estructura apuntada por p_direccion; lg indica el tamaño de la dirección.

Para una utilización en un dominio particular, el puntero p_direccion apunta a una zona cuya estructura es la de una dirección en ese dominio (sockaddr_un para el dominio AF_UNIX y sockaddr_in para el dominio AF_INET).

El socket nombrado después de realizarse con éxito la primitiva (valor de retorno 0), puede ser identificado por cualquier proceso por medio de la dirección que se le ha dado sin necesidad de poseer un descriptor. Un cierto número de primitivas específicas permiten explotar estas direcciones.
Las causas de error (valor devuelto -1) de petición de conexión son múltiples: descriptor incorrecto, dirección incorrecta, inaccesible o ya utilizada, o un socket ya conectado a una dirección.

Un proceso puede disponer de un descriptor de un socket conectado o enlazado a una dirección pero sin saber cuál es (si la conexión ha sido realizada por otro proceso y el proceso ha heredado el descriptor o si la conexión ha sido realizada sin especificar el número de puerto). La primitiva permite recuperar la dirección relacionada con el socket del descriptor sock:

```
int getsockname (sock, p_adr, p_lg)
int sock; /* descriptor del socket */
struct sockaddr *p_adr; /* puntero a la zona de dirección */
int *p_lg; /* puntero a la longitud de la dirección */
```

Cuando se llama a esta primitiva, el tercer parámetro se utiliza como dato y como resultado:

- En la llamada, *p_lg tiene como valor el tamaño de la zona reservada a la dirección p_adr para recuperar la dirección del socket.

- En el retorno de la llamada, tiene como valor el tamaño efectivo de la dirección. El valor de retorno de la primitiva es 0 o -1 según si la llamada ha tenido éxito o no.

2.3.6. La primitiva Connect

Después de la creación de un socket, un proceso hace la llamada connect para establecer una conexión activa con otro proceso remoto, es decir, abre un circuito virtual entre dos sockets, el cual permite intercambios bidireccionales:

```
int connect (sock, p_adr, lgadr)
int sock; /* descriptor del socket local */
```

```
struct sockaddr *p_adr; /* dirección del socket remoto */
int lgadr; /* longitud dirección remota */
```

El socket correspondiente al descriptor sock será conectado o enlazado a una dirección local. Connect intenta contactar con el ordenador remoto con el objeto de realizar una conexión entre el socket remoto y el socket local especificado en sock.

La conexión puede establecerse (y la primitiva connect devuelve el valor 0) si se cumplen las siguientes condiciones:

a) Los parámetros son "localmente" correctos.

b) La dirección *p_adr se asocia a un socket del tipo SOCK_STREAM en el mismo dominio que el socket local de descriptor sock y un proceso (servidor) tiene solicitado escuchar sobre este socket (por una llamada a listen).

c) La dirección *p_adr no está utilizada por otra conexión.

d) El archivo de conexiones pendientes del socket distante o remoto no está lleno.

En caso de éxito (valor de retorno 0 de la primitiva connect), el socket local sock está conectado con un nuevo socket y la conexión está pendiente hasta que el proceso llamado tenga conocimiento de ella a través de la primitiva accept. Sin embargo, el proceso que llama puede comenzar a escribir o a leer del socket.

En el caso de que no se cumpla alguna de las tres primeras condiciones, el valor devuelto por la primitiva es -1, indicando error.

El comportamiento de la primitiva es particular si no se cumple la condición d:

- Si el socket es de modo bloqueante, el proceso se bloquea. La petición de conexión se repite durante un cierto tiempo; si al cabo de este lapso de tiempo la conexión no se ha podido establecer, el proceso es despertado.

- Si el socket es de modo no bloqueante, la vuelta es inmediata. Sin embargo, la petición de conexión no se abandona en seguida (se repite durante el mismo lapso de tiempo). Finalmente, en el caso de que se realice una nueva solicitud de conexión del mismo socket (por una llamada a connect) antes de que se haya abandonado la petición anterior, la vuelta de la primitiva es igualmente inmediata.

2.3.7. La Primitiva Write

A través de esta primitiva, dos procesos pueden intercambiar información a través de una conexión TCP. En el caso de un cliente y un servidor, el cliente realizaría con estas primitivas peticiones, y el servidor les daría réplica.

Esta primitiva requiere tres argumentos: el descriptor de un socket al cual deben ser enviados los datos, la dirección de los datos a ser enviados y la longitud de los datos.

```
int write (sock, msg, lg)
int sock; /* descriptor del socket local */
```

```
char *msg; /* dirección de memoria del mensaje a enviar */
int lg; /* longitud del mensaje */
```

Normalmente, para facilitar la comunicación, la primitiva write no manda la información directamente sobre el socket, sino que lo hace sobre un buffer perteneciente al kernel del Sistema operativo, lo que permite que la aplicación siga su ejecución mientras se transmiten los datos a través de la red. En el caso de que el buffer estuviese lleno, se bloquearía la aplicación hasta que los datos del mismo pudiesen ser mandados por la red.

2.3.8. La Primitiva Read.

De la misma forma que con la primitiva write los procesos mandaban información a través de la red, con la primitiva read, los procesos esperan información procedente de otros procesos que desean comunicar con ellos. El uso de una primitiva write por parte del proceso llamante, implica que para que el proceso receptor pueda recibir lo que ha escrito el primero, necesita ejecutar la primitiva read, de forma que el proceso receptor pueda leer los datos de una conexión TCP.

Al igual que la primitiva write, esta primitiva requiere también tres argumentos: el descriptor del socket a usar, la dirección del mensaje a enviar y la longitud de dicho mensaje.

```
int write (sock, msg, lg)
int sock; /* descriptor del socket local */
char *msg; /* dirección de memoria del mensaje a enviar */
int lg; /* longitud del mensaje */
```

Cuando llegan datos, read los copia del socket y los ubica en el buffer del área de usuario. En el caso de que no haya llegado ningún dato, read bloquea la aplicación hasta que llegue algún dato al socket. Si llega más información de la que cabe en el buffer, read solo extrae lo que tenga cabida en el buffer, y en el caso de que hayan llegado menos datos que espacio en el buffer, read los copia todos y devuelve el número de bytes leídos.

2.3.9. La primitiva Listen

Cuando un socket es creado, no se encuentra en estado pasivo (por ejemplo, preparado para ser usado por un servidor) ni activo (por ejemplo, para ser usado por un cliente), hasta que la aplicación tome una acción determinada.

Los servidores en modo conectado realizan la llamada listen para colocar un socket en modo pasivo (pasándolo como argumento), y prepararlo para futuras conexiones entrantes que tendrán lugar, es decir, no hará nada hasta que llegue una conexión. Para llamar con éxito a esta primitiva (con valor de retorno 0), el proceso debe disponer de un descriptor de socket del tipo SOCK_STREAM.

La mayoría de los servidores consisten en un bucle infinito que acepta cada conexión entrante que le llega, la maneja, y después sigue en el bucle a la espera de otras conexiones. Si en el momento en el que el servidor atiende una petición (conexión) no llega ninguna otra conexión, no existe problema, pero este si existe en el momento en que el servidor está atendiendo una determinada conexión, y en ese instante llega otra. Para asegurar de que esta nueva conexión no se pierda, a la primitiva listen se le pasa un argumento que le indica al Sistema operativo que encole la petición de conexión para un socket recibida. Así pues, los dos argumentos que se les pasa a la llamada

listen son el socket que debe ser ubicado en modo pasivo, y el tamaño de la cola que va a ser usada por dicho socket.

```
int listen (sock, nb)
int sock; /* descriptor de socket */
int nb; /* número máximo de peticiones de conexión pendiente */
```

2.3.10. La Primitiva Accept

Esta primitiva permite extraer una conexión pendiente de la cola asociada a un socket para la cual se ha realizado una llamada a listen. De esta manera, el Sistema toma conocimiento de un enlace realizado.

```
int accept (sock, p_adr, p_lgadr)
int sock; /* descriptor del socket */
struct sockaddr *p_adr; /* dirección del socket conectado */
int *p_lgadr; /* puntero al tamaño de la zona reservada a p_adr */
```

Una vez extraída la petición de conexión, accept crea un nuevo socket con las mismas propiedades que sock y reserva (que se devuelve como resultado de la función) un nuevo descriptor de fichero para él. El socket de servicio creado se enlaza a un nuevo puerto (tomado del conjunto de puertos no reservados).

A la vuelta, también se recupera memoria en la zona apuntada por p_adr, la dirección del socket del cliente con el cual se ha restablecido la conexión. El valor de *p_lgadr se modifica: si en la llamada era el tamaño de la zona reservada para guardar la dirección, en el retorno da el tamaño efectivo de la dirección.

En el caso en el que no exista ninguna conexión pendiente, el proceso se bloquea hasta que exista una (o hasta que llegue una señal) a menos que el socket esté en modo no bloqueante.

2.3.11. Sumario de Primitivas

Hasta ahora, hemos visto las primitivas más importantes usadas con sockets. Pero hay muchas más, por lo que pasamos a hacer un pequeño esquema en el que se muestran todas las primitivas existentes usadas con TCP:

Primitivas de Sockets	
Primitiva	**Función**
Socket	Crea un descriptor para que sea usado en la comunicación.
Connect	Conexión con un cliente remoto.
Write	Manda datos a través de una conexión.
Read	Lee los datos entrantes de una conexión.
Close	Termina la conexión y elimina el descriptor.
Bind	Vincula una dirección local IP y un puerto de protocolo a un socket.
Listen	Pone el socket en modo pasivo y establece el número de conexiones TCP entrantes que el Sistema puede encolar.
Accept	Acepta la siguiente conexión entrante.
Recv	Recibe el siguiente datagrama entrante.
Recvmsg	Recibe el siguiente datagrama entrante (variación de recv).
Recvfrom	Recibe el siguiente datagrama entrante y graba la dirección fuente.
Send	Envía un datagrama.
Select	Informa de sockets listos para escribir o leer de ellos.
Sendmsg	Envía un datagrama (variación de send).
Sendto	Envía un datagrama, usualmente a una dirección previamente grabada.
Shutdown	Termina un conexión TCP en una o ambas direcciones.
Getpeername	Después de la llegada de una conexión, obtiene la dirección completa de la máquina remota desde un socket.

Getsockopt	Obtiene las opciones actuales de un socket.
Setsockopt	Cambia la opción de un socket.

2.4. Propiedades, Métodos y Eventos de WinSock

Una vez que tenemos el WinSock control en nuestra barra de controles en Visual Basic ya podemos comenzar a ver las propiedades, eventos y métodos más importantes del control. Para agregarlo manualmente ir a Proyecto> Componentes> y luego seleccionar WinSock Control y Aceptar. Como mencionamos anteriormente este control no es visible en tiempo de ejecución.
Primero abrimos un proyecto (EXE Estándar) y colocamos en control en cualquier parte del formulario. Vamos a comenzar por ver las propiedades, estas pueden ser puestas en tiempo de diseño como también en tiempo de ejecución.

2.4.1. Lista de Propiedades más Importantes

***LocalIP*:** Devuelve la dirección IP de la máquina local en el formato de cadena con puntos de dirección IP (xxx.xxx.xxx.xxx).

***LocalHostName*:** Devuelve el nombre de la máquina local.

***RemoteHost*:** Establece el equipo remoto al que se quiere solicitar la conexión.

***LocalPort*:** Establece el puerto que se quiere dejar a la escucha.

***RemotePort*:** Establece el número del puerto remoto al que se quiere conectar.

***State*:** Verifica si el Control WinSock esta siendo utilizado o no.

Estas son algunas de las propiedades más importantes, y a continuación la sintaxis de cada propiedad.

Objeto.Propiedad = Valor

Donde Objeto va el nombre del Control WinSock, el nombre predeterminado cuando lo incluimos en alguna aplicación es “WinSock1”. Luego le sigue la propiedad que deseamos asignar y finalmente el valor que la misma tomará.

Entonces por ejemplo si queremos probar la propiedad *LocalIP* debemos seguir el ejemplo 1.

Ejemplo 1

Crear un Proyecto (EXE Estándar) y agregar el WinSock Control. Luego agregar una etiqueta vacía, es decir un Label. Después introducimos el siguiente código.

```
Private Sub Form_Load()
   Label1.caption = WinSock1.LocalIP
End Sub
```

Este simple ejemplo nos da de forma rápida nuestro IP, aunque no estemos conectados a Internet el IP aparece igual, solo que siempre va a tomar el valor: 127.0.0.1

Ahora que sabemos manejar las propiedades podemos seguir con los Métodos. A continuación la lista de algunos de los Métodos más importantes del Control WinSock.

2.4.2. Lista de Métodos más Importantes

***Accept*:** Sólo para las aplicaciones de servidor TCP. Este método se utiliza para aceptar una conexión entrante cuando se está tratando un evento ConnectionRequest.

***GetData*:** Recupera el bloque actual de datos y lo almacena en una variable de tipo Variant.

***Listen*:** Crea un socket y lo establece a modo de escucha.

***SendData*:** Envía datos a un equipo remoto.

2.4.3. Lista de Eventos más Importantes

***ConnectionRequest*:** Se produce cuando el equipo remoto solicita una conexión. Sin este evento no se puede llevar a cabo la conexión.

***Connect*:** Se produce cuando el equipo local se conecta al equipo remoto y se establece una conexión.

***Close*:** Se produce cuando el equipo remoto cierra la conexión. Las aplicaciones deben usar el método Close para cerrar correctamente una conexión TCP.

***DataArrival*:** Se produce cuando llegan nuevos datos. Este evento es importante, ya que debemos hacer al con la información que llega.

La sintaxis de los métodos y eventos es igual a la sintaxis de las propiedades, por lo cual no vamos a hacer referencia a ella.

2.5. Programando la Primera Aplicación Cliente/Servidor

Conociendo las propiedades, métodos y eventos del Control WinSock podemos pasar a la engorrosa labor de la programación.

Para poder programar la siguiente aplicación necesitan tener el Control WinSock en el formulario, eso siempre es fundamental para que el programa ande.

Para entender el correcto funcionamiento del protocolo TCP/IP vamos a empezar por programar la aplicación Servidor a la cual luego se conectará el Cliente.

Comenzamos por crear un proyecto nuevo (EXE estándar) para el ***Servidor***, y agregamos la siguiente lista de controles al formulario principal. La ubicación de dichos controles es a gusto del programador, siempre tratando de que el usuario final este a gusto con el producto y que se pueda manejar libremente sin problemas por el entorno del mismo.

- WinSock Control.

- 2 cajas de texto (TextBox).

- 2 botones.

A continuación hace falta que cambiemos algunas propiedades de los controles, debajo la lista de controles con las respectivas propiedades a cambiar.

Control *(nombre predeterminado)*	**Propiedad *(nuevo valor)***
WinSock1	LocalPort = 888
Text1	Text =
Text2	Text =
Command1	Caption = “Escuchar”
Command2	Caption = “Enviar”

Para que el ejemplo funcione a la perfección conviene que seguir la ubicación de los controles como esta indicado en la Figura 2.6.

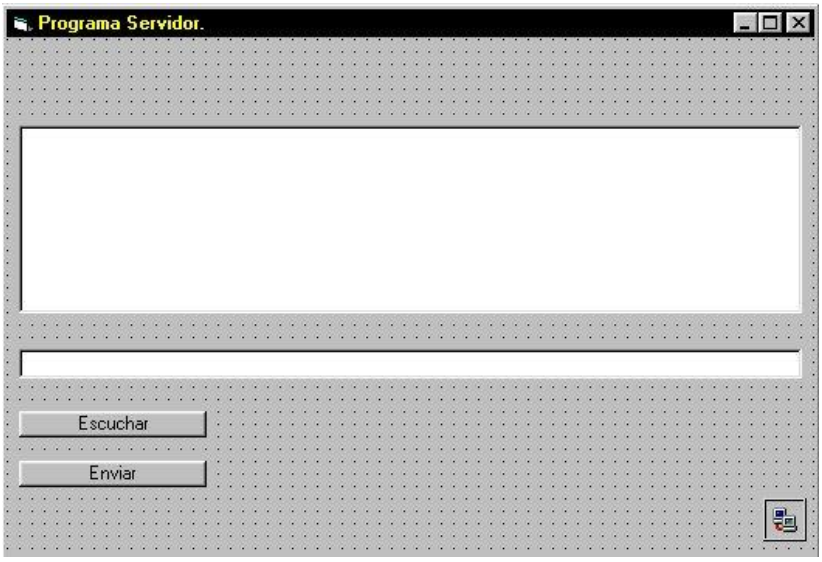


Figura 2.6. Programa Servidor.

Una vez hecho esto podemos empezar a tipear código. El sangrado del programa es una cuestión de entendimiento para el programador, algunos recurren a éste como otros no, eso también queda a criterio del que programa.
En el Evento Click del Command1 incluimos el siguiente código; (sólo lo que esta en **NEGRITA**, el resto es en modo de ayuda, ya que aparece cuando se hace doble click en algún control).

```
Private Sub Command1_Click()
  Winsock1.Listen
End Sub
```

Esto hace que el Control WinSock empiece a funcionar, escuchando el puerto que se indicó en las propiedades de dicho control. Este puerto es el 888. Ahora si realizamos todo a la perfección el puerto 888 esta siendo vigilado para aceptar conexiones remotas.

Luego en el Evento DataArrival del WinSock;

```
Private Sub Winsock1_DataArrival(By Val bytes Total As Long)
  Dim dates As String
  Winsock1.GetData dates
  Text1.Text = Text1.Text + dates
End Sub
```

Datos queda transformada en una variable de cadena, y WinSock almacena los datos que recibe del Cliente en el buffer y luego ingresan a la variable datos, dicha variable mostrará su contenido en el control TextBox (Text1).

En el evento ConnectionRequest;

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
  Winsock1.Close
  Winsock1.Accept requestID
End Sub
```

Este evento es muy importante, permite aceptar la petición de conexión. Sin este evento el resto del programa no tendría efecto.

En el evento Click del command2;

```
Private Sub Command2_Click()
  Dim enviar As String
  enviar = Text2.Text
  Winsock1.SendData enviar
End Sub
```

Esto permite enviar el texto que se introduzca en el TextBox número 2.

Por ahora este es un simple programa Servidor, lo que hace es: designar un puerto, dejarlo a la escucha para aceptar conexiones, si se realiza una petición de conexión aceptarla, y por último enviar datos al Cliente y recibir los datos que éste mande.

Para seguir programando el ***Cliente*** hace falta crear un nuevo proyecto y en el formulario principal incluir la siguiente lista de controles:

- WinSock Control.
- 3 cajas de texto (TextBox).

- 2 botones.

Como lo hicimos anteriormente hace falta cambiar algunas propiedades. Debajo la lista de controles con las respectivas propiedades para cambiar.

Control *(nombre predeterminado)*	**Propiedad *(nuevo valor)***
WinSock1	RemotePort = 888
Text1	Text =
Text2	Text =
Text3	Text =
Command1	Caption = “Conectar”
Command2	Caption = “Enviar”

Para tener una referencia de cómo situar los controles conviene seguir la Figura 2.7.

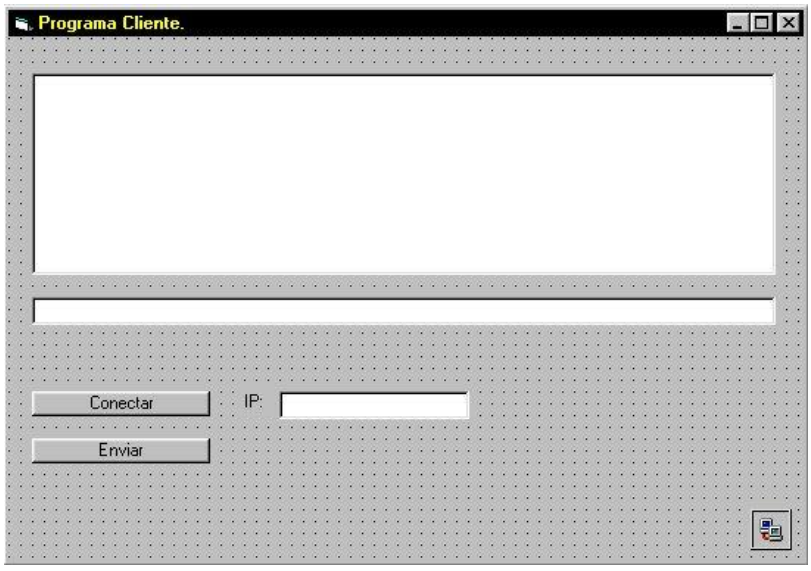


Figura 2.7. Programa Cliente.

En el método del command1;

```
Private Sub Command1_Click()
  Winsock1.RemoteHost = Text3.Text
  Winsock1.Connect
End Sub
```

El evento connect permite conectar al programa servidor que esta esperando la solicitud, este evento requiere un parámetro fundamental, el IP o nombre de host el cual es introducido previamente a la conexión en el cuadro de texto número 3 (Text3).

En el evento DataArrival del WinSock Control;

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
  Dim datos As String
  Winsock1.GetData datos
  Text1.Text = Text1.Text + datos
End Sub
```

Esto permite a la aplicación (a través de WinSock) recibir información del servidor y mostrarla en pantalla.

En el método del command2;

```
Private Sub Command2_Click()
  Dim enviar As String
  enviar = Text2.Text
  Winsock1.SendData enviar
End Sub
```

Estas instrucciones son necesarias para enviar información al servidor.

Este ejemplo del primer programa Cliente / Servidor es muy simple, para utilizarlo al máximo es necesario por ejemplo poner las propiedades de los TextBox en Multiline, lo que hace que si los datos recibidos exceden el tamaño del TextBox estos datos vayan directo a la línea de abajo.

3.1. Análisis y Diseño del Sistema Propuesto

Para la creación de Sistema Cliente/Servidor aplicado a puntos de venta a través de Socktes se ha utilizado el Análisis Estructurado que introduce el uso de las herramientas de documentación gráficas para producir un tipo diferente de especificación funcional, el cual nos brindara los cimientos que serán la base importante para la construcción del Sistema.

La lógica del software esta estrechamente relacionado con el conocimiento del personal de trabajo de la Ferretería San Agustín.

Mediante el Modelo Entidad Relación nos permite representar con claridad las limitantes de los datos, ya está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y linguísticas.

Para el modelamiento del Sistema se ha utilizado el diagrama de flujo de datos (DFD), que es una herramienta que permite visualizar un Sistema como una red de procesos funcionales. Siendo éste, una de las herramientas más comúnmente usadas, sobre todo por Sistemas operacionales en los cuales las funciones del Sistema son de gran importancia y son más complejos que los datos que éste maneja.

3.2. Recopilación de Datos

Para la recopilación de la información, así como para su tratamiento se necesito la colaboración de las siguientes personas:

- Dueño de la empresa.

- Secretaria.
- Bodegueros.
- Vendedores.
- Clientes.

3.3. Especificación de Requerimientos

3.3.1. Requerimientos de Software

Los requerimientos de Software indispensables para la elaboración del Sistema son los siguientes:

- Sistema Operativo Windows 2000 Advanced Server (Servidor).
- Sistema Operativo Windows 98 o superior (Cliente).
- Microsoft Visual Basic 6.0.
- SQL SERVER 2000.
- Crystal Report 8.5.
- Microsoft Office.

3.3.2. Requerimientos de Hardware

Entre los requerimientos de hardware tenemos los siguientes:

- Computador Pentium III o superior.
- Monitor SVGA de 14'' con una resolución de 800 x 600 píxeles o mayor.
- 128 Mb en RAM mínimo.
- Mouse.
- CD-ROM.
- Tarjeta de Red 10/100.
- Impresora matricial en cada punto de venta.

3.4. Desarrollo del Sistema

3.4.1. Requerimientos del Usuario con el Sistema

Dentro de los requerimientos del usuario con el sistema tenemos los siguientes:

- Diseñar una interfaz agradable y de fácil uso.
- Tener un almacenamiento confiable de la información.

- Información en línea.
- Manejo rápido, óptimo e inteligente.
- Visualizar reportes.
- Impresión de reportes.
- Dominio del Sistema por teclado y mouse.
- Seguridades del Sistema.

3.4.2. Análisis de Datos Necesarios para el Sistema

3.4.3. Identificación de Datos

El Sistema trabaja con los siguientes datos que identifican a:

- Artículos.
- Ajuste de inventario.
- Bancos.
- Caja.
- Clientes.

- Cuentas por cobrar.
- Cuentas por pagar.
- Empresa.
- Facturas.
- Inicio de caja.
- Gastos de caja.
- Grupos.
- Pedidos.
- Proveedores.
- Subgrupos.
- Usuarios.
- Vendedores.

3.4.3.1. Análisis de datos

Para mayor facilidad y entendimiento de los datos se ha realizado la siguiente definición de datos:

Definición de Datos			
Nombre	**Tipo**	**Longitud**	**Descripción**
Ced	nvarchar	10	Identifica el número de cedula de un Cliente
Telef	nvarchar	10	Representa un número de teléfono
Fax	nvarchar	10	Representa un número de fax
Ruc	nvarchar	13	Representa un número de Ruc de un Proveedor
Dato	varchar	25	Representa una cadena de caracteres
Uni	varchar	25	Representa el nombre de una unidad de presentación
Categ	varchar	25	Representa el nombre de una Categoría de un artículo
Direcc	varchar	40	Identifica una dirección
Nom	varchar	40	Representa el nombre de un Articulo, Proveedor, Cliente
Id	char	7	Representa un valor que se utiliza como identificador de código en las bodegas, ajustes
Cant	int	4	Identifica un valor entero que se asigna a una cantidad
Desc	float	8	Representa un valor que se utiliza para realizar un descuento
Prec	money	8	Identifica un valor de precio o costo que se asigna a una artículo en una compra o en un pedido
Logo	image	16	Identifica un Logotipo ya sea de un articulo o empresa
Fecha	fecha		Representa una fecha en que se realiza una Compra, Pedido, Ajustes o una modificación de un Articulo
Coment	varchar	50	Representa un comentario, una frase de texto
Status	bit	1	Representa un valor 0 o 1

En base a la identificación y análisis de datos se genero la base de datos con la cual trabajara el Sistema.

3.5. Normalización de la Base de Datos

La normalización es una técnica que se utiliza para crear relaciones lógicas apropiadas entre tablas de una base de datos.

3.5.1. Tercera Forma Normal

La regla de la Tercera Forma Normal señala que hay que eliminar y separar cualquier dato que no sea clave.

Esto le da más flexibilidad y previene errores de lógica cuando inserta o borra registros. Cada columna en la tabla está identificada de manera única por la clave, y no hay datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

A continuación se presenta las tablas expresadas en su Tercera Forma Normal:

T_Ajustes (Aj_id, ***Us_id***, Aj_fecha, Aj_tipo, Aj_comen, Bodega, Status**)**

T_Articulos (Ar_id, ***Gr_id***, ***Su_id***, ***Pr_id***, Ar_nom, Ar_cant, Ar_max, Ar_min, Ar_prec1, Ar_prec2, Ar_prec3, Ar_uni1, Ar_uni2, Ar_uni3, Ar_descu, Ar_costoact, Ar_ultcosto, Ar_nota1, Ar_nota2, Ar_nota3, Ar_nota3, Ar_logo, B1, B2, B3, B4, Ar_status, Iva, Ult_precv, cant1, cant2, cant3, cant4**)**

T_Bancos (Ba_id, Ba_nom, Ba_cuent**)**

T_Bod_ajus (*Ar_id*, *Aj_id*, Ar_nom, cant, prec, total**)**

T_Caja (Ca_id, Ca_nom, status, nota**)**

T_Ciecaja (*Ca_id*, *Us_id*, Id, fecha, hora, efectivo, cheque, crédito, tarjeta, inicio, gastos, cobros, devoluciones**)**

T_Clientes (Cl_id, ***Tcl_id***, Cl_nom, Cl_apell, Cl_ruc, Cl_dir, Cl_telef, Cl_email, Cl_ciudad, Contacto, Teléfono, Fax, Fpago, cuenta_banco, Ba_id, Cl_limcred, Cl_saldo, Cl_disponible, Cl_ultcomp, Descu, aviso, status, nota, exento, acogido**)**

T_Cobros (*Ca_id*, *Cxc_id*, *Ba_id*, num_cuotas, forma _ pago, valor, Vuelto, cheque, fecha_depo, fecha _ cobro**)**

T_CuentasXC (*Cxc_id*, *Fa_id*, fecha_emision, valor_inicial, pago, saldo_actual, status**)**

T_CuentasXP (*Cxp_id*, *Pe_id*, fecha_emision, valor_inicial, pago, saldo_actual, referenciadocu, status, retencion**)**

T_DPedidos (*Pe_id*, *Ar_id*, nom, cant, costo, descu, precio_u, total, iva**)**

T_Dfactura (*Fa_id*, *Ar_id*, nom, cant, num_prec, precio_U, iva, descu, total**)**

T_Empresa (id, nom, dir, tel, fax, email, ruc, ciu, provi, repre, rep_ced, rep_dir, rep_telef, logotipo, nota1, nota2, nota3**)**

T_Estadistica (*Ar_id*, num_ventas, uni_vendidas, num_compras, uni_compradas, ult_preciov, ult_venta, ult_compra**)**

T_Factura (Fa_id, ***Cl_id***, ***Us_id***, ***Ca_id***, Fa_tipo, Fa_fecha, Fa_fecha1, Bodega, status, recargo, total**)**

T_Gascaja (*Ca_id*, *Us_id*, num, fecha, hora, monto, nota**)**

T_Grupo (Gr_id, Gr_nom**)**

T_Inicaja (*Ca_id*, *Us_id*, num, fecha, hora, monto, nota)

T_Pagos (*Cxp_id*, *Ba_id*, num_cuotas, forma_pago, valor, Num_cheque, Fecha_deposito, fecha_pago)

T_Pedidos (Pe_id, *Pr_id*, *Us_id*, Pe_tipo, Pe_fecha, Pe_hora, Bodega, numdocp, status, recargo, total)

T_Proveedores (Pr_id, *Tpr_id*, Pr_nom, Pr_ruc, Pr_direcc, Pr_telef, Pr_fax, Pr_email, Pr_ciudad, Pr_provincia, Pr_contacto, Pr_cont_telef, Pr_cont_email, Pr_cuenta_banc, formapago, saldo, ultimacompra, Ba_id, descuento, beneficio, excentoiva, acogidote, nota, aviso, status, tpagado)

T_Subgrupo (Su_id, *Gr_id*, Su_nom)

T_TipoCliente (Tcl, Tc_nom, Nota)

T_Tproveedor (Tpr_id, nombre, Nota)

T_Usuarios (Us_id, Us_nom, Us_clave, perfil, status)

T_ocacionales (Fa_id, Id, nom, apell, dir, ruc, fecha)

3.6. Diagrama Entidad Relación

3.7. Diseño de Tablas

Nombre de la tabla: T_Ajustes

Nombre Campo	***Tipo***
Aj_id	*cod*
Aj_fecha	*fecha*
Aj_tipo	*datos*
Aj_comen	*datos*
Bodega	*id*
Status	*status*
Us_id	*cod*

Nombre de la tabla: T_Articulos

Nombre Campo	***Tipo***
Ar_id	*cod*
Ar_nom	*nom*
Ar_cant	*id*
Ar_max	*int*
Ar_min	*int*
Ar_prec1	*prec*
Ar_prec2	*prec*
Ar_prec3	*prec*
Ar_uni1	*unid*
Ar_uni2	*unid*
Ar_uni3	*unid*
Gr_id	*id*
Su_id	*id*

Ar_codultprov	*cod*
Ar_descu	*int*
Ar_costoact	*prec*
Ar_ultcosto	*prec*
Ar_nota1	*datos*
Ar_nota2	*datos*
Ar_nota3	*datos*
Ar_nota3	*datos*
Ar_logo	*image*
B1	*int*
B2	*int*
B3	*int*
B4	*int*
Ar_status	*bit*
Iva	*int*
Ult_precv	*prec*
cant1	*int*
cant2	*int*
cant3	*int*
cant4	*int*

Nombre de la tabla: T_Bancos

Nombre Campo	***Tipo***
Ba_id	*id*
Ba_nom	*nom*
Ba_cuent	*nom*

Nombre de la tabla: T_Bod_ajus

Nombre Campo	***Tipo***
Ar_id	*cod*
Ar_nom	*nom*
cant	*int*
prec	*cant*
total	*cant*
Aj_id	*cod*

Nombre de la tabla: T_Caja

Nombre Campo	***Tipo***
Ca_id	*cod*
Ca_nom	*nom*
status	*status*
nota	*nom*

Nombre de la tabla: T_Ciecaja

Nombre Campo	***Tipo***
Id	*int*
fecha	*fecha*
hora	*fecha*
ca_id	*cod*
efectivo	*prec*
cheque	*prec*
credito	*prec*
tarjeta	*prec*
inicio	*prec*
gastos	*prec*
cobros	*prec*
devoluciones	*prec*
us_id	*cod*

Nombre de la tabla: T_Clientes

Nombre Campo	***Tipo***
Cl_id	*cod*
Cl_nom	*nom*
Cl_apell	*nom*
Cl_ruc	*ruc*
Cl_dir	*direcc*
Cl_telef	*telef*
Cl_email	*datos*
Cl_ciudad	*datos*
Tc_id	*cod*
Contacto	*nom*

Teléfono	*telef*
Fax	*telef*
Fpago	*datos*
cuentabanco	*datos*
Ba_id	*datos*
Cl_limcred	*int*
Cl_saldo	*prec*
Cl_disponible	*int*
Cl_ultcomp	*fecha*
Descu	*desc*
aviso	*status*
status	*status*
nota	*datos*
exento	*status*
acogido	*status*

Nombre de la tabla: T_Cobros

Nombre Campo	***Tipo***
num_cuotas	*int*
forma_pago	*nom*
valor	*prec*
Vuelto	*prec*
Ba_id	*id*
cheque	*nom*
fecha_depo	*fecha*
fecha_cobro	*fecha*
Ca_id	*cod*
CXC_id	*cod*

Nombre de la tabla: T_CuentasXC

Nombre Campo	***Tipo***
Cxc_id	*cod*
Fa_id	*cod*
fecha_emision	*fecha*
valor_inicial	*prec*
pago	*prec*
saldo_actual	*prec*
status	*status*

Nombre de la tabla: T_CuentasXP

Nombre Campo	***Tipo***
Cxp_id	*cod*
Pe_id	*cod*
fecha_emision	*fecha*
valor_inicial	*prec*
pago	*prec*
saldo_actual	*prec*
referenciadocu	*datos*
status	*status*
retencion	*desc*

Nombre de la tabla: T_DPedidos

Nombre Campo	***Tipo***
Ar_id	*cod*
nom	*nom*
cant	*int*
costo	*prec*
descu	*int*
precio_u	*prec*
total	*prec*
Pe_id	*cod*
iva	*int*

Nombre de la tabla: T_Dfactura

Nombre Campo	***Tipo***
Ar_id	*cod*
nom	*nom*
cant	*cant*
num_prec	*int*
precio_U	*prec*
iva	*int*
descu	*int*
total	*prec*
Fa_id	*cod*

Nombre de la tabla: T_Empresa

Nombre Campo	***Tipo***
id	*id*
nom	*nom*
dir	*direc*
tel	*telef*
fax	*telef*
email	*direc*
ruc	*nom*
ciu	*direcc*
provi	*direcc*
repre	*nom*
rep_ced	*ced*
rep_dir	*direc*
rep_telef	*telef*
logotipo	*image*
nota1	*datos*
nota2	*datos*
nota3	*datos*

Nombre de la tabla: T_Estadistica

Nombre Campo	***Tipo***
ar_id	*cod*
num_ventas	*int*
uni_vendidas	*int*
num_compras	*int*
uni_compradas	*int*
ult_preciov	*int*
ult_venta	*fecha*
ult_compra	*fecha*

Nombre de la tabla: T_Factura

Nombre Campo	***Tipo***
Fa_id	*cod*
Fa_tipo	*datos*
Fa_fecha	*fecha*
Fa_fecha1	*fecha*
Cl_id	*cod*
Us_id	*cod*
Bodega	*id*
Ca_id	*cod*
status	*status*
recargo	*prec*
total	*prec*

Nombre de la tabla: T_Gascaja

Nombre Campo	***Tipo***
num	*int*
fecha	*fecha*
hora	*fecha*
Ca_id	*cod*
monto	*prec*
nota	*datos*
Us_id	*cod*

Nombre de la tabla: T_Grupo

Nombre Campo	***Tipo***
Gr_id	*id*
Gr_nom	*nom*

Nombre de la tabla: T_Inicaja

Nombre Campo	***Tipo***
num	*int*
fecha	*fecha*
Hora	*fecha*
Ca_id	*cod*
monto	*prec*
nota	*datos*
Us_id	*cod*

Nombre de la tabla: T_Pagos

Nombre Campo	***Tipo***
num_cuotas	*int*
forma_pago	*datos*
valor	*prec*
Ba_id	*id*
Num_cheque	*datos*
Fecha_deposito	*fecha*
fecha_pago	*fecha*
Cxp_id	*cod*

Nombre de la tabla: T_Pedidos

Nombre Campo	***Tipo***
Pe_id	*cod*
Pe_tipo	*datos*
Pe_fecha	*fecha*
Pe_hora	*fecha*
Pr_id	*cod*
Us_id	*cod*
Bodega	*id*
numdocp	*id*
status	*status*
recargo	*prec*
total	*prec*

Nombre de la tabla: T_Proveedores

Nombre Campo	***Tipo***
Pr_id	*cod*
Pr_nom	*nom*
Pr_ruc	*ruc*
Pr_direcc	*direcc*
Pr_telef	*telef*
Pr_fax	*telef*
Pr_email	*datos*
Pr_ciudad	*datos*
Pr_provincia	*nom*
Pr_contacto	*nom*
Pr_cont_telef	*telef*
Pr_cont_email	*nom*
Pr_cuenta_banc	*datos*
id	*cod*
formapago	*datos*
saldo	*prec*
ultimacompra	*fecha*
Ba_id	*datos*
descuento	*int*
beneficio	*int*
excentoiva	*status*
acogidore	*status*
nota	*datos*
aviso	*status*
status	*status*
tpagado	*prec*

Nombre de la tabla: T_Subgrupo

Nombre Campo	***Tipo***
Su_id	*id*
Su_nom	*nom*
Gr_id	*id*

Nombre de la tabla: T_TipoCliente

Nombre Campo	***Tipo***
Tc_id	*cod*
Tc_nom	*nom*

Nombre de la tabla: T_Tproveedor

Nombre Campo	***Tipo***
Id	*cod*
nombre	*nom*
Nota	*datos*

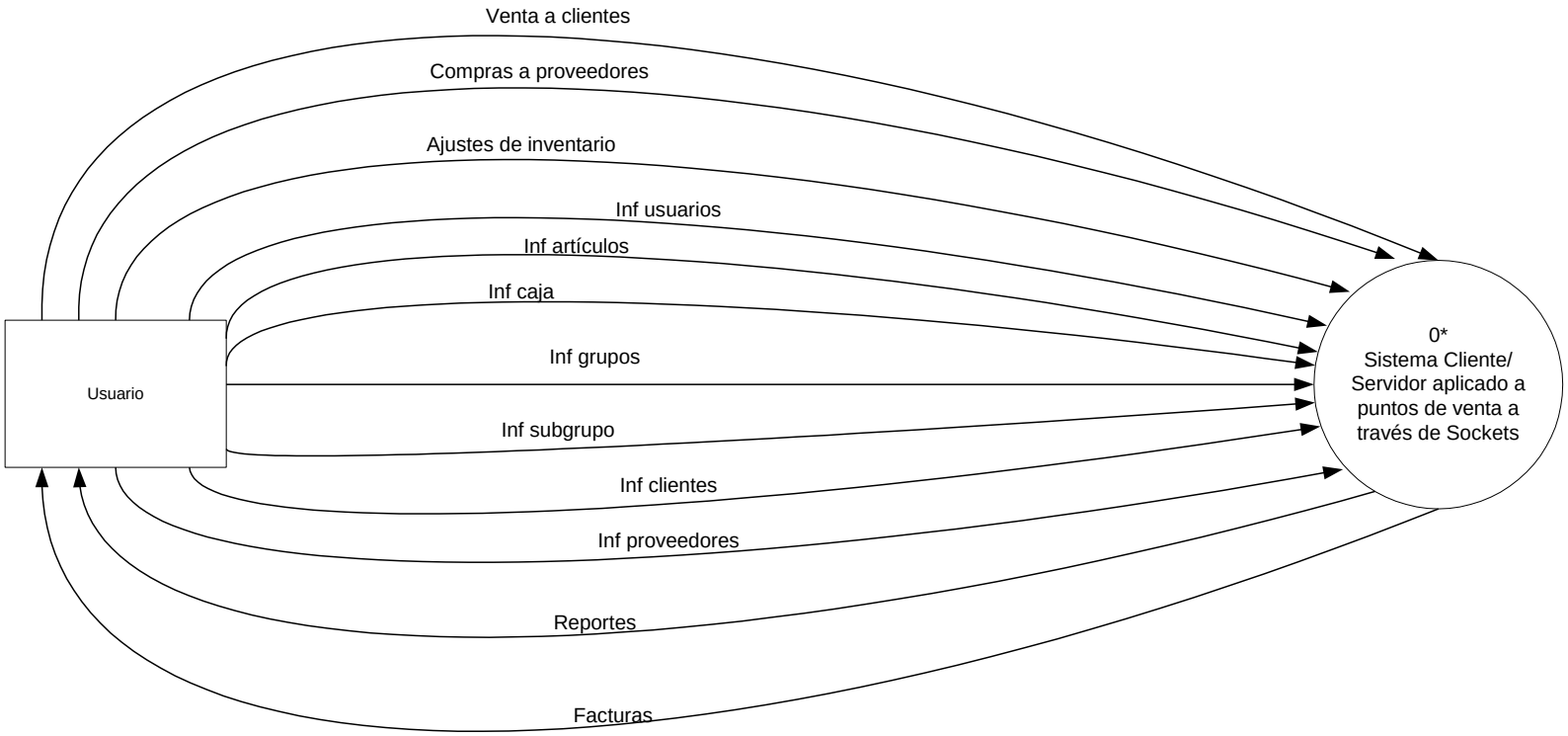
Nombre de la tabla: T_Usuarios

Nombre Campo	***Tipo***
Us_id	*cod*
Us_nom	*nom*
Us_clave	*id*
perfil	*datos*
status	*status*

Nombre de la tabla: T_ocacionales

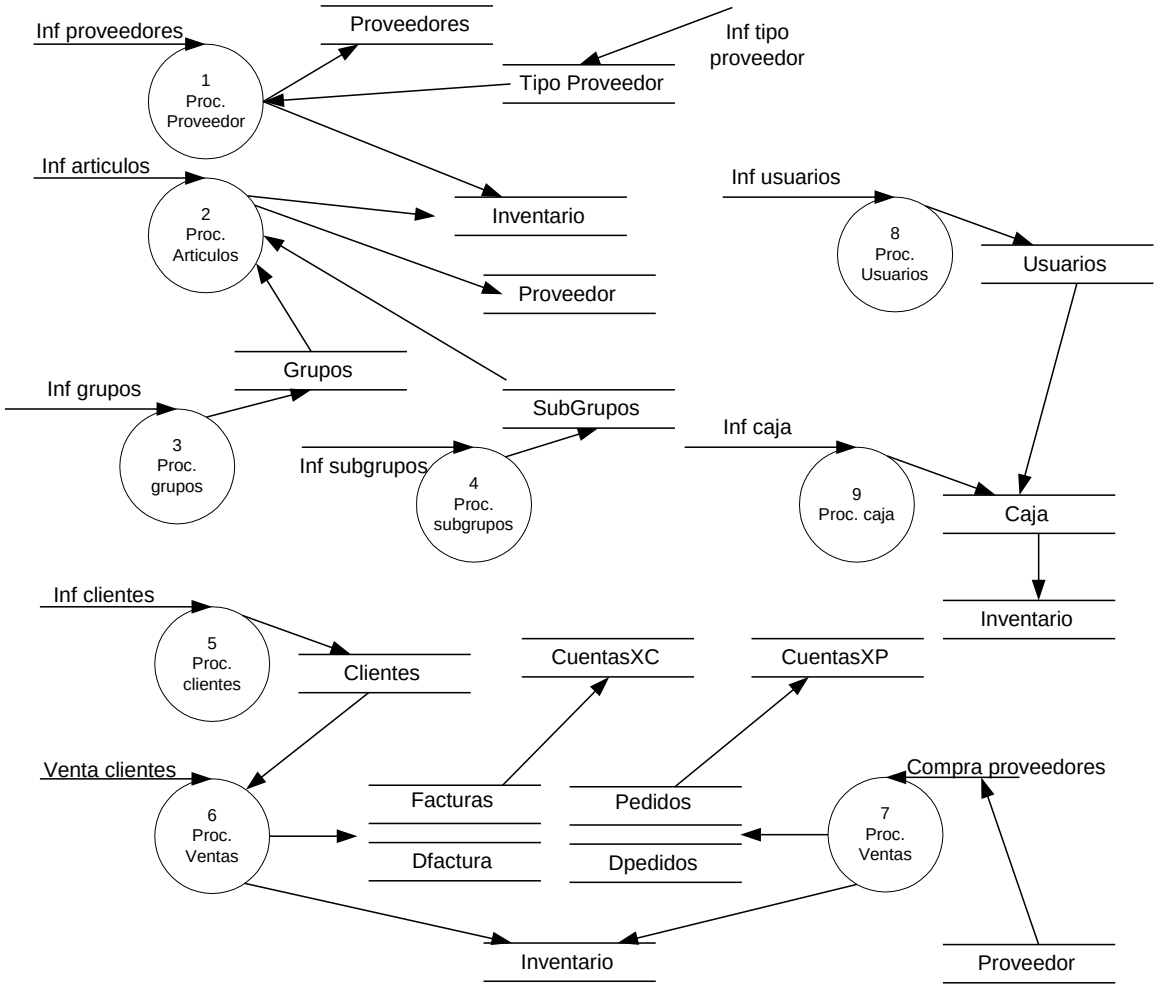
Nombre Campo	***Tipo***
Id	*int*
nom	*nom*
apell	*nom*
dir	*nom*
ruc	*ruc*
fa_id	*cod*
fecha	*fecha*

3.8. Diagrama de Contexto



3.9. Diagrama de Flujo de Datos

NIVEL 1



NIVEL 2

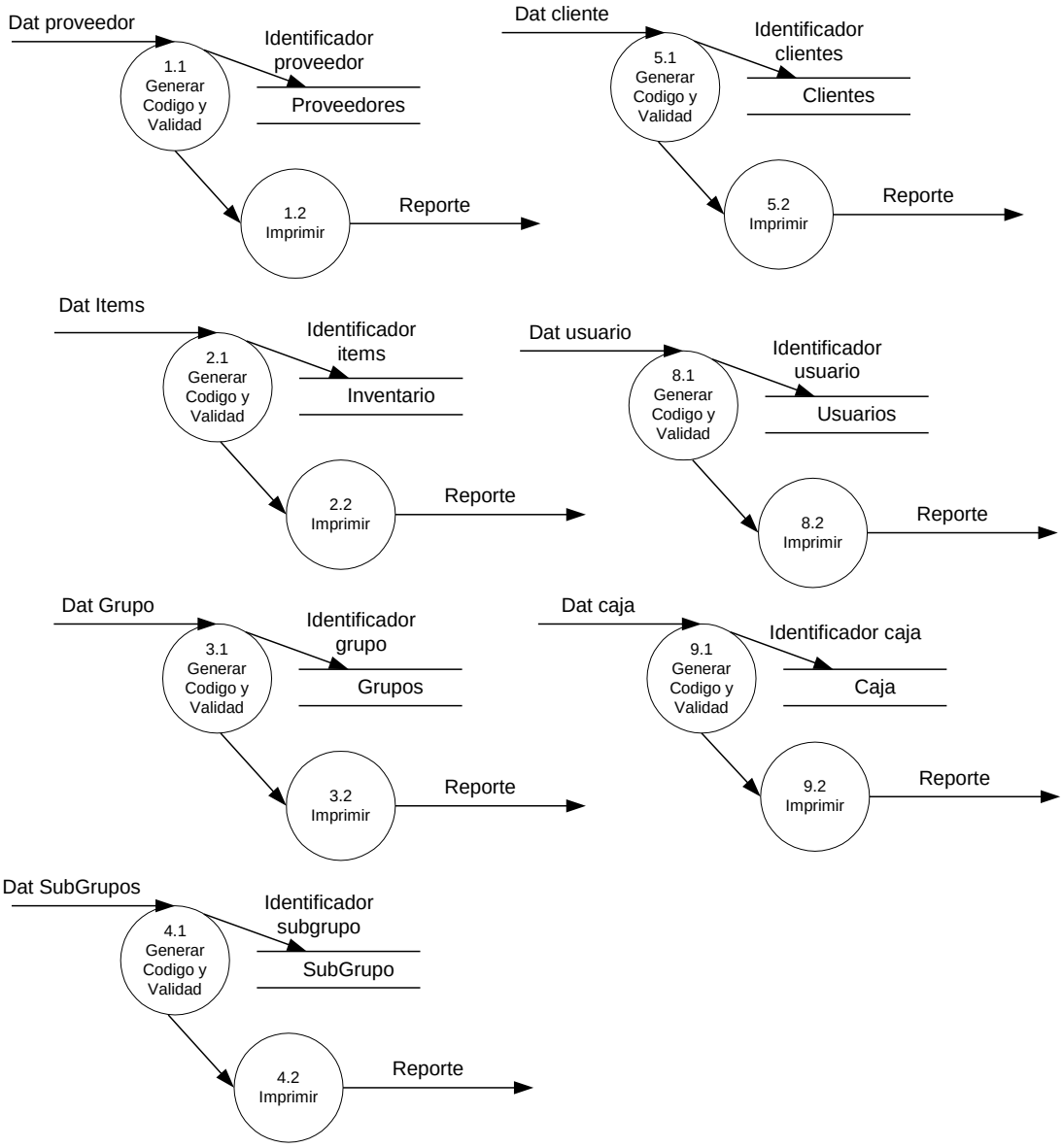


DIAGRAMA 1

Generación de código, validación e impresión

NIVEL 2

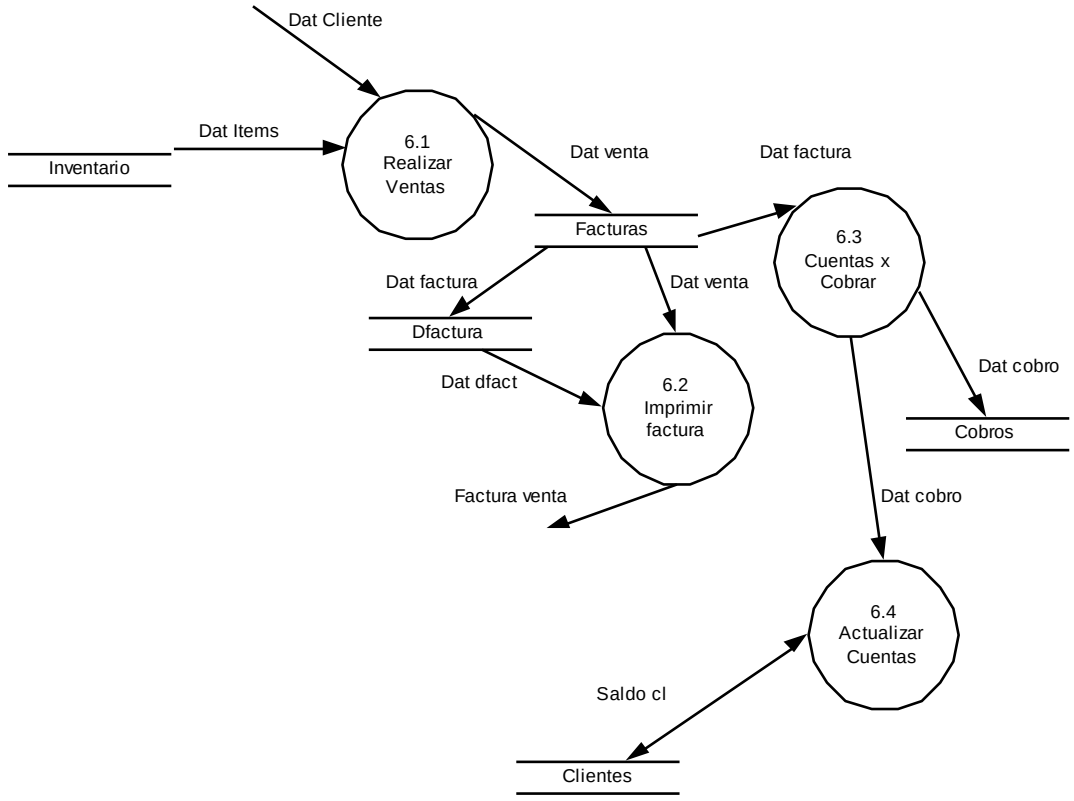


DIAGRAMA 2

Ventas

NIVEL 2

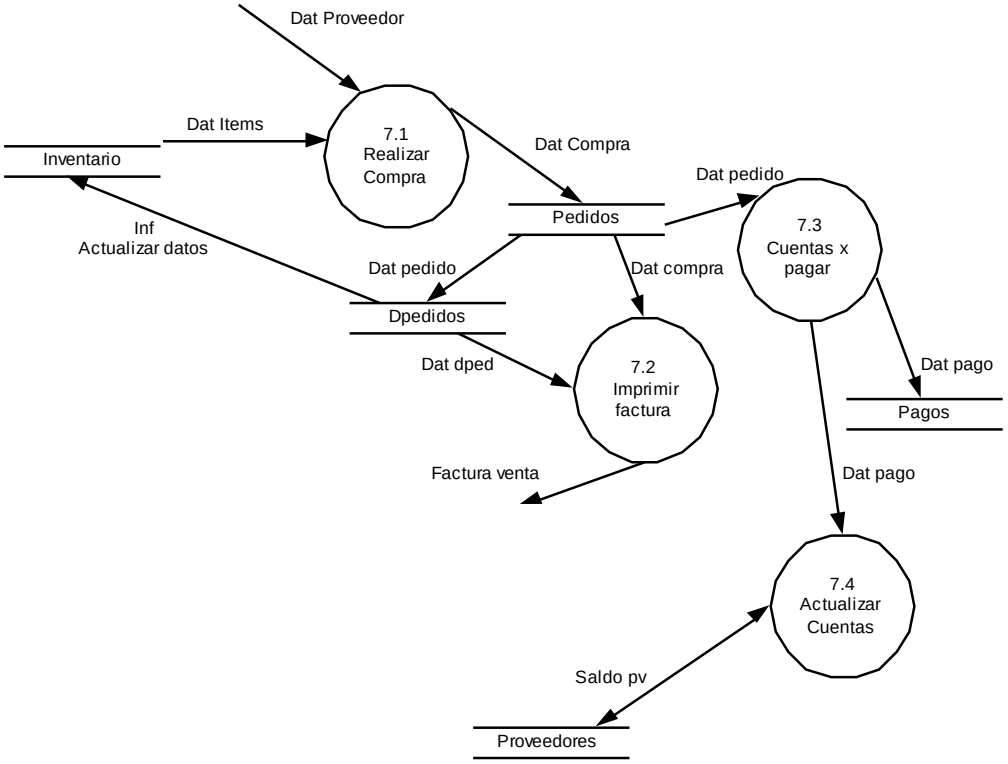


DIAGRAMA 3

Compras

3.10.Diccionario de Datos

3.10.1. Flujo de Datos

Nombre del Flujo: Inf proveedores
Información:
(pr_nom)

Nombre del Flujo: Inf tipo proveedores
Información:
(id, nombre)

Nombre del Flujo: Inf artículos
Información:
(ar_nom, gr_id, su_id)

Nombre del Flujo: Inf usuarios
Información:
(us_nom, us_clave, perfil)

Nombre del Flujo: Inf grupos
Información:
(gr_nom)

Nombre del Flujo: Inf subgrupo
Información:
(su_nom, gr_id)

Nombre del Flujo: Inf caja
Información:
(ca_id, ca_nom)

Nombre del Flujo: Inf clientes
Información:
(cl_nom, tc_id)

Nombre del Flujo: Dat proveedor
Información:
(pr_id)

Nombre del Flujo: Dat cliente
Información:
(cl_id)

Nombre del Flujo: Dat items
Información:
(ar_id)

Nombre del Flujo: Dat usuario
Información:
(us_id)

Nombre del Flujo: Dat grupo
Información:
(gr_id)

Nombre del Flujo: Dat caja
Información:
(ca_id)

Nombre del Flujo: Dat subgrupos
Información:
(su_id)

Nombre del Flujo: Dat venta
Información:
(fa_id, fecha, cl_id)

Nombre del Flujo: Dat factura
Información:
(fa_id, fecha, ca_id)

Nombre del Flujo: Dat dfactura
Información:
(fa_id, ar_id)

Nombre del Flujo: Dat cobro
Información:
(cxc_id, fa_id)

Nombre del Flujo: Dat saldo cl
Información:
(cxc_id, fa_id, cl_id)

Nombre del Flujo: Dat compra
Información:
(pe_id, fecha, pr_id)

Nombre del Flujo: Dat pedido
Información:
(pe_id, fecha, ca_id)

Nombre del Flujo: Dat dpedido
Información:
(pe_id, ar_id)

Nombre del Flujo: Dat pago
Información:
(cxp_id, pe_id)

Nombre del Flujo: Dat saldo pv
Información:
(cxp_id, pe_id, pr_id)

4.1. Diseño de Entradas y Salidas

4.2. Diseño de Entradas

De la calidad de entradas al Sistema depende la calidad de salidas del mismo para lo cual se ha tomado en cuenta los siguientes aspectos:

- Eficacia.
- Precisión.
- Facilidad de uso.
- Consistencia.
- Sencillez y atracción.
- Uniformidad de colores.
- Posibilidad de cancelar.

Para el ingreso al Sistema el usuario debe elegir la base de datos con la cual va a trabajar, indicar el perfil que este tiene, ingresar su nombre y clave de ingreso respectivamente, el usuario tiene tres oportunidades para su ingreso al Sistema si los datos proporcionados son incorrectos, la pantalla de ingreso de usuarios se ilustra en la Figura 4.1.

Figura 4.1 Ingreso de Usuarios.

Una vez verificado el nombre de usuario, su clave y perfil se realiza el ingreso al Sistema, donde se presenta un menú con todas las opciones a las cuales el usuario tiene acceso. El menú del Sistema se ilustra en la Figura 4.2.

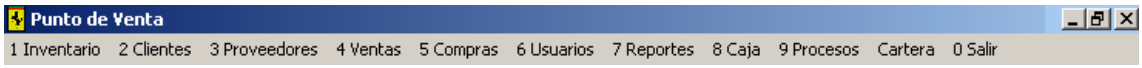


Figura 4.2. Menú Principal del Sistema.

4.2.1. Breve Presentación de las Opciones del Menú Principal

En el menú Inventario tenemos, las opciones de Grupo, Grupo-Subgrupo, Ítems, Ajuste Inventario, como se ilustra en la Figura 4.3.

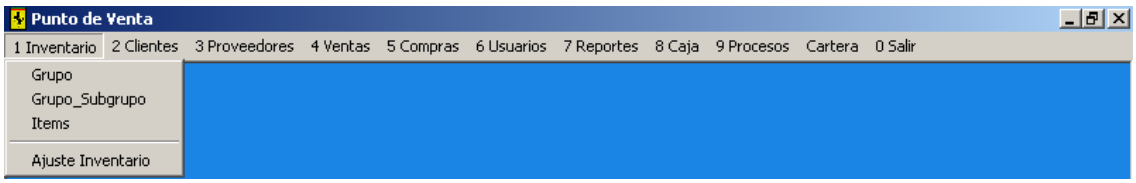


Figura 4.3. Opciones del Menú Inventario.

Para un mejor control los ítems (artículos) disponibles en la ferretería San Agustín se los ha agrupado en Grupos, como se ilustra en la Figura 4.4.

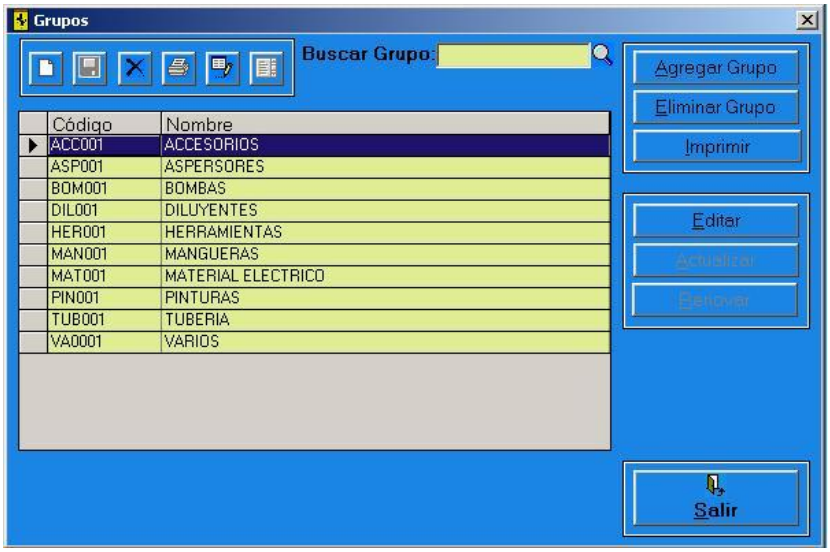


Figura 4.4. Grupo de Ítems.

Cada Grupo puede tener uno o más Subgrupos de acuerdo a las necesidades, la opción grupo-subgrupo se ilustra en la Figura 4.5.

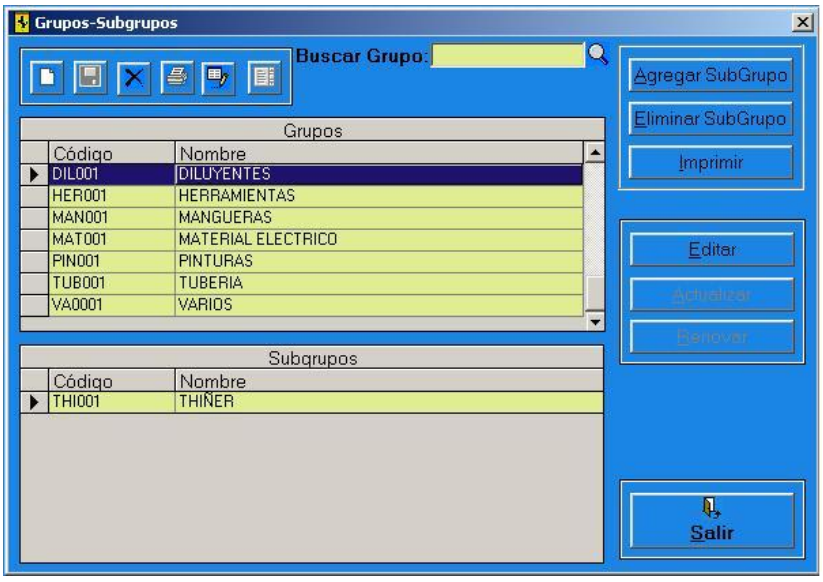


Figura 4.5. Grupos-Subgrupos.

En el submenú ítems se presenta información individual de cada ítem, se ha utilizado una presentación maestro-detalle para una mejor visualización y tratamiento de la información, como se ilustra en la Figura 4.6.

Figura 4.6. Presentación de Ítems.

En el submenú Ajuste del inventario podemos realizar egresos e ingresos, refiérase a la Figura 4.7.

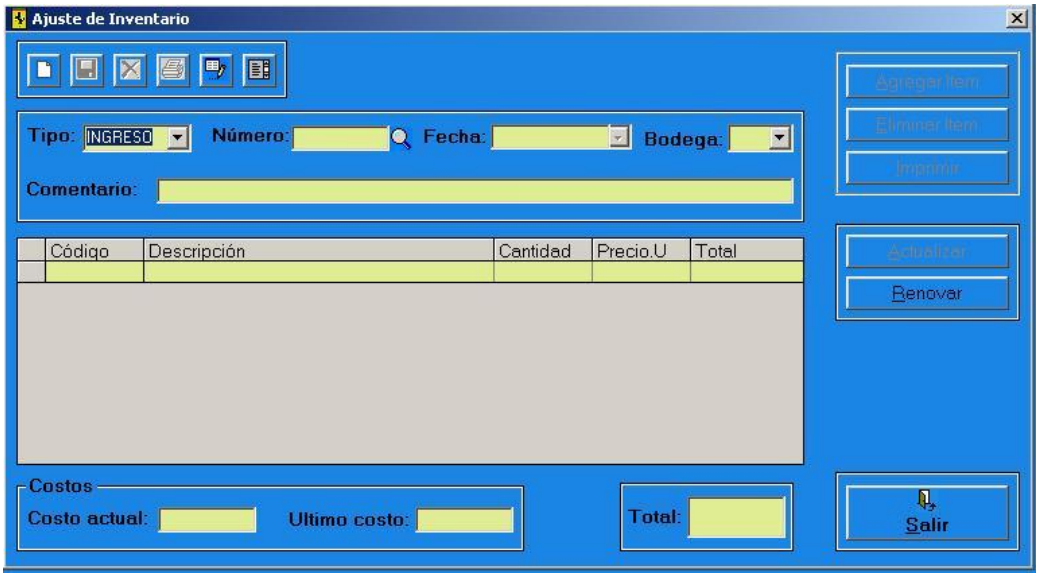


Figura 45.7. Ajuste de Inventario.

Menú clientes se presenta información individual de cada cliente, como se ilustra en la Figura 4.8.

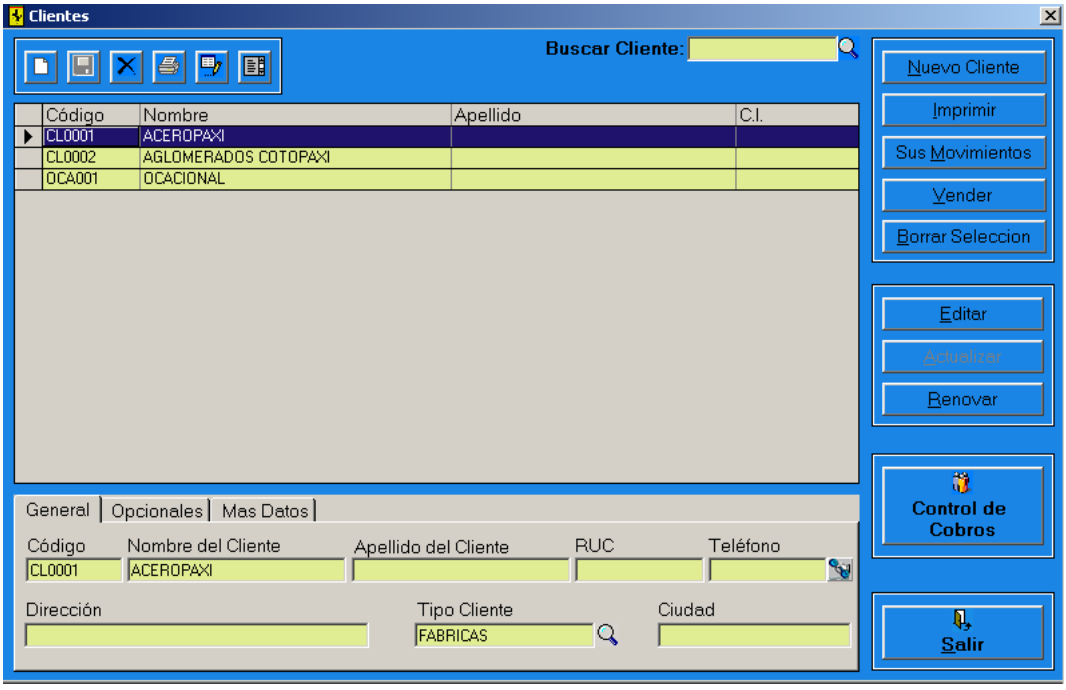


Figura 4.8. Menú Clientes.

Menú Proveedores se presenta información individual de cada proveedor, como se ilustra en la Figura 4.9.

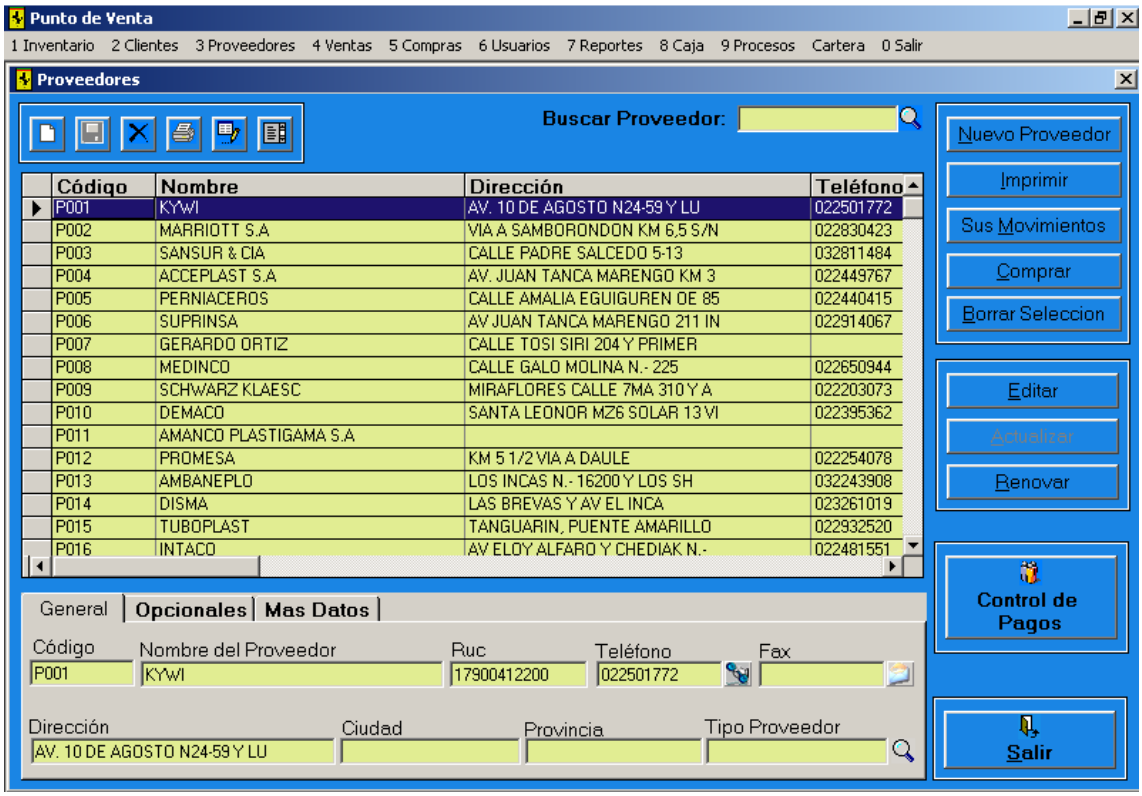


Figura 4.9. Menú Proveedores.

El menú ventas se ilustra en la Figura 4.10.

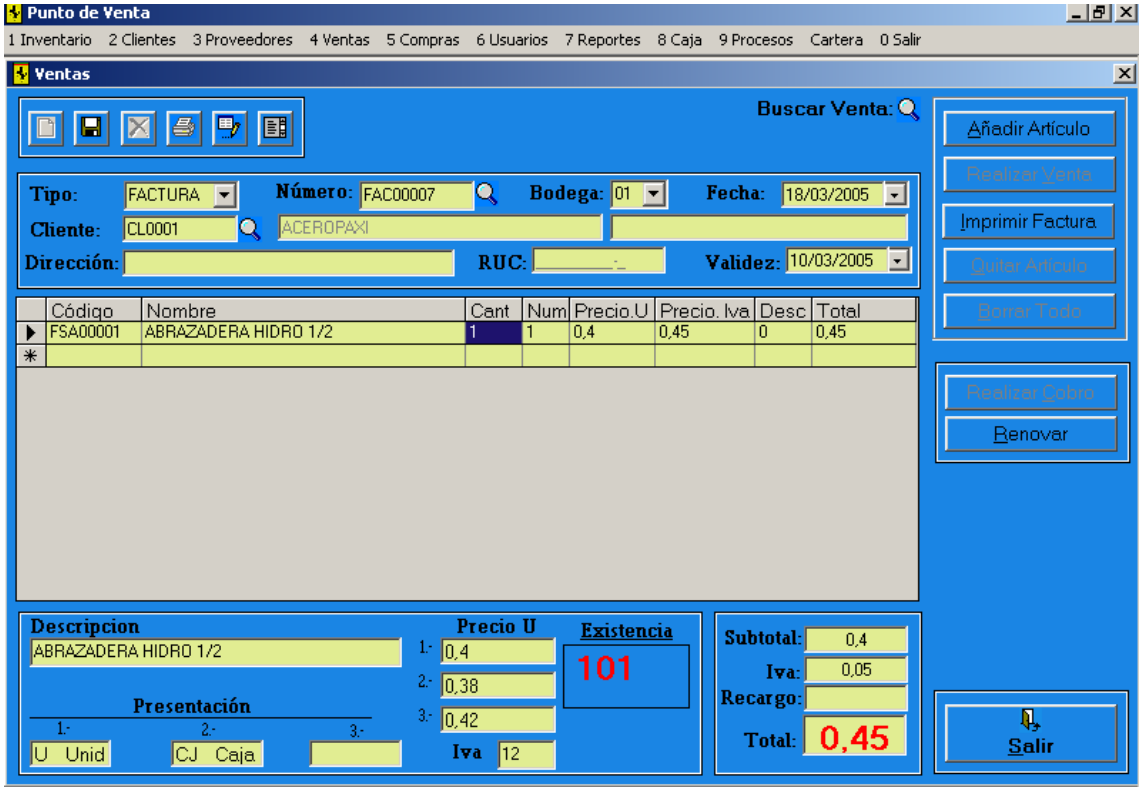


Figura 4.10. Menú Ventas.

El menú compras se ilustra en la Figura 4.11.

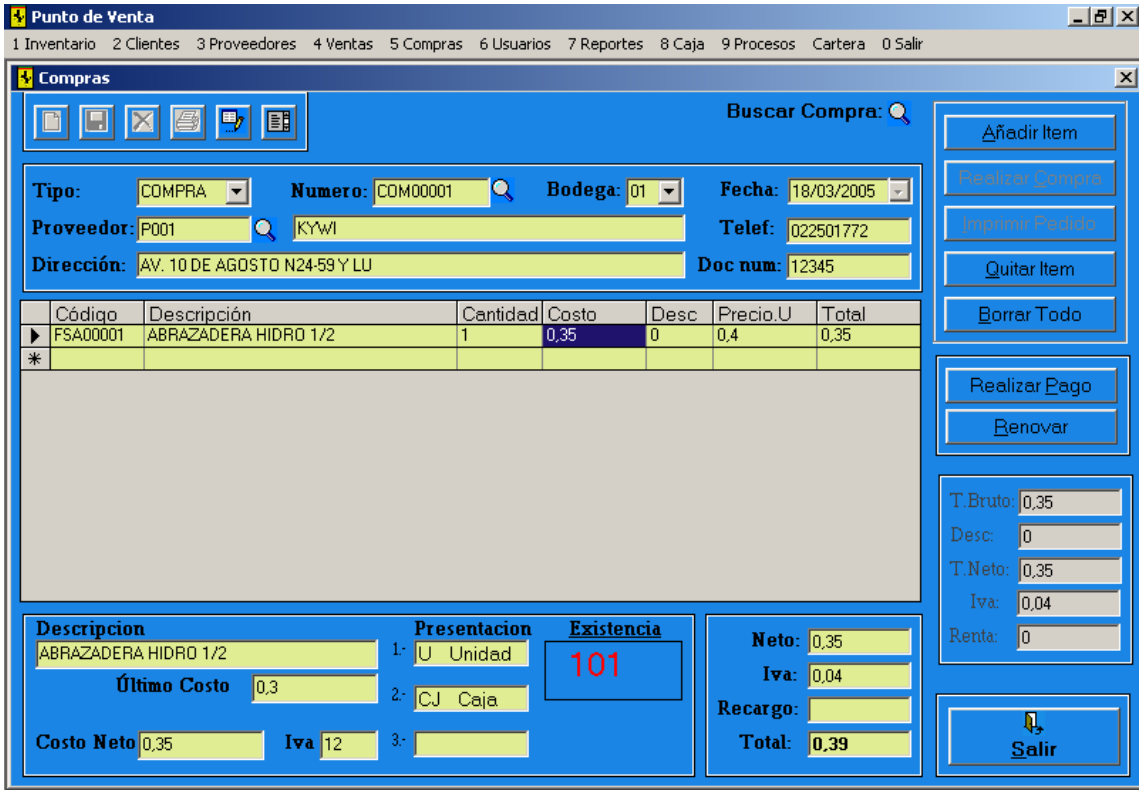


Figura 4.11. Menú Compras.

El menú usuarios y su respectivo submenú se ilustra en la Figura 4.12.

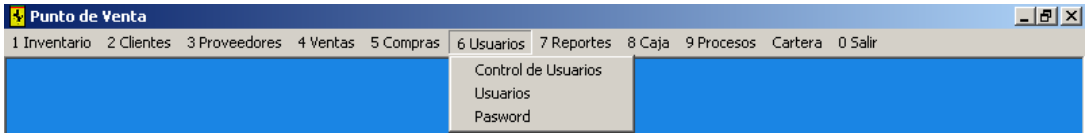


Figura 4.12. Menú Usuarios y Submenú.

El submenú control de usuarios nos permite saber que usuario esta conectado, como se ilustra en la Figura 4.13.

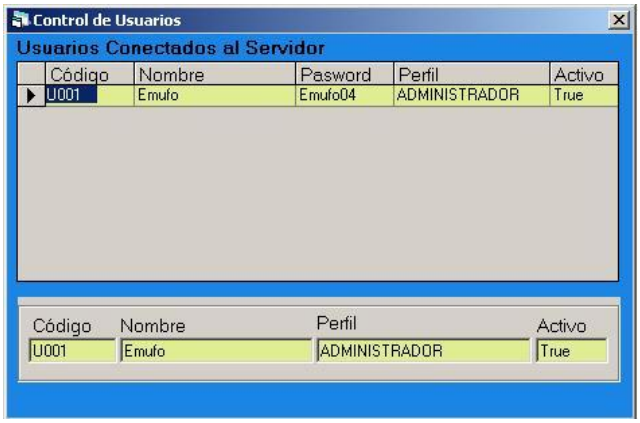


Figura 4.13. Control de Usuarios.

El submenú usuarios se ilustra en la Figura 4.14.

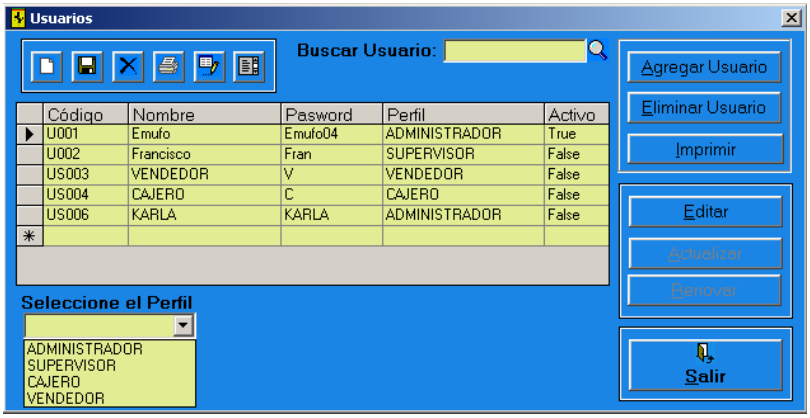


Figura 4.14. Creación de Usuarios.

El submenú pasword se ilustra en la Figura 4.15.

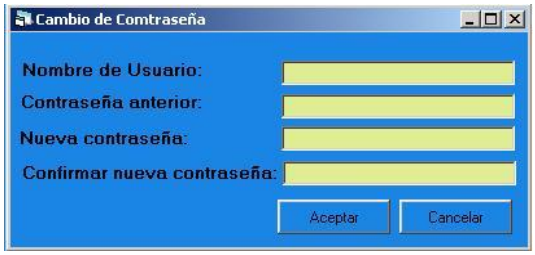


Figura 4.15. Cambio de Pasword.

El menú reportes y su respectivo submenú se ilustra en la Figura 4.16.

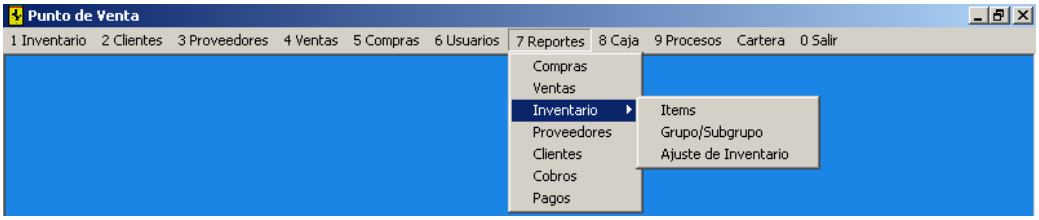


Figura 4.16. Menú Reportes.

Cada submenú del menú reportes tienen es siguiente esquema, refiérase a la Figura 4.17.

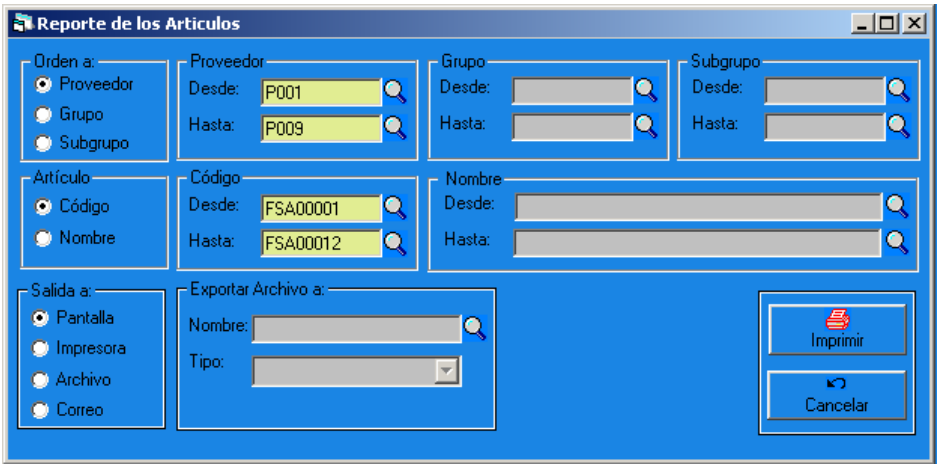


Figura 4.17. Reporte de Artículos.

El menú Caja y su respectivo submenú, se ilustra en la Figura 4.18.

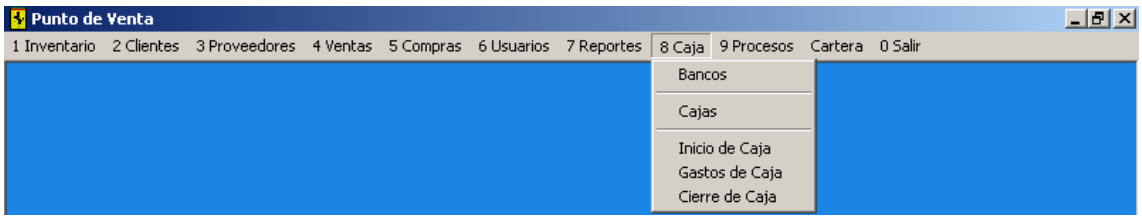


Figura 4.18. Menú Caja.

El submenú bancos se ilustra en la Figura 4.19.

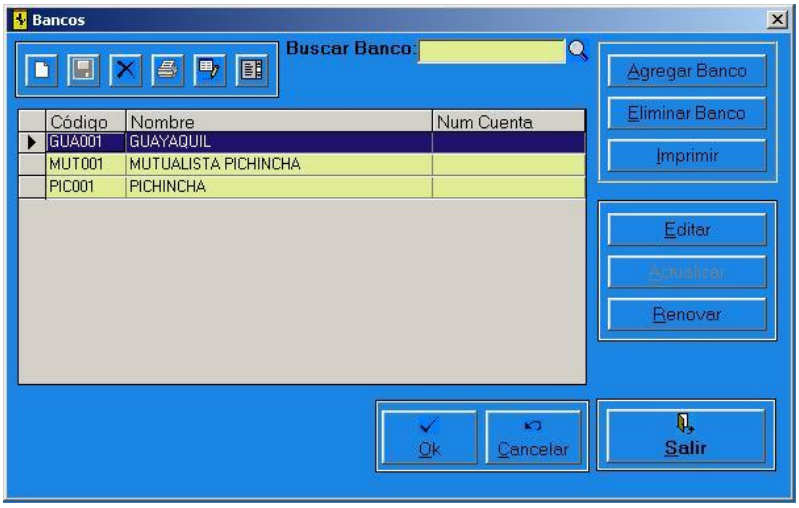


Figura 4.19. Submenú Bancos.

El submenú cajas se ilustra en la Figura 4.20.

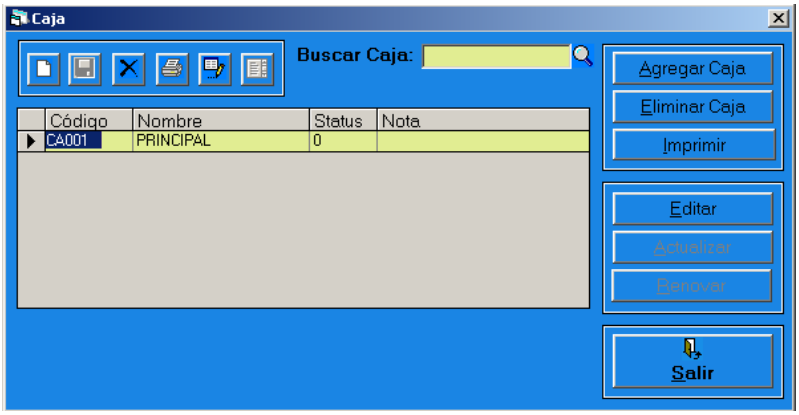


Figura 4.20. Submenú Cajas.

El submenú inicio de caja se ilustra en la Figura 4.21.

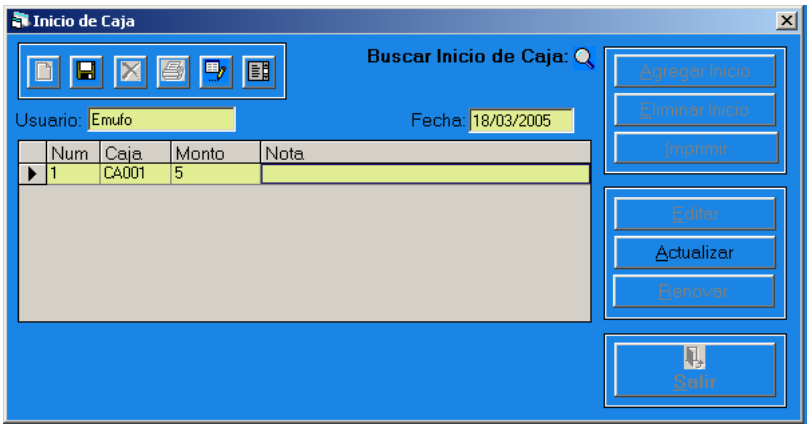


Figura 4.21. Submenú Inicio de Caja.

El submenú gastos de caja se ilustra en la Figura 4.22.

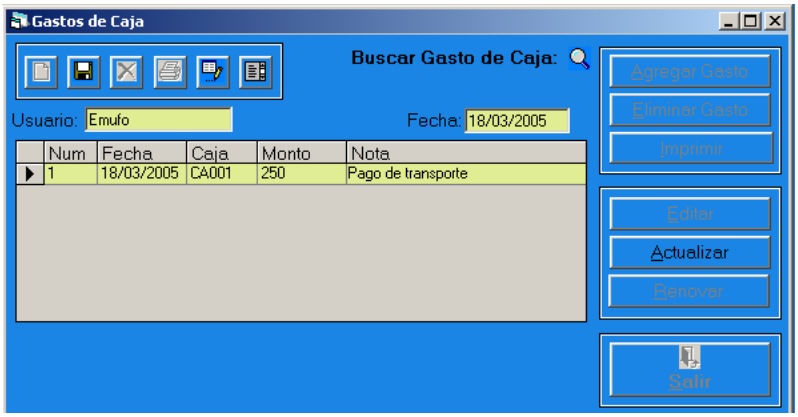


Figura 4.22. Submenú Gastos de Caja.

El submenú cierre de caja se ilustra en la Figura 4.23.

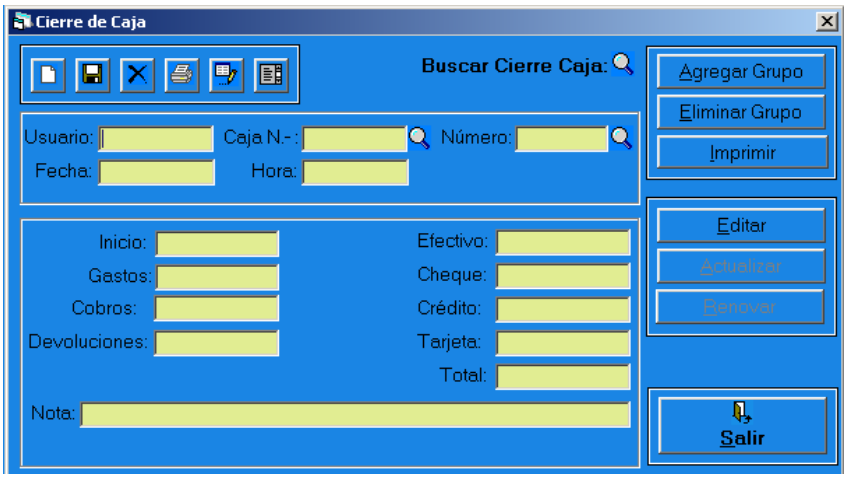


Figura 4.23. Submenú Cierre de Caja.

El menú procesos y su respectivo submenú se ilustran en la Figura 4.24.

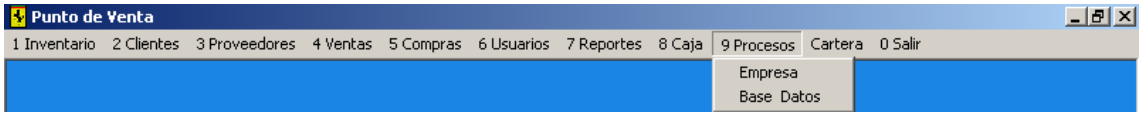


Figura 4.24. Menú Procesos y Submenú.

El submenú datos empresa se ilustra en la Figura 4.25.

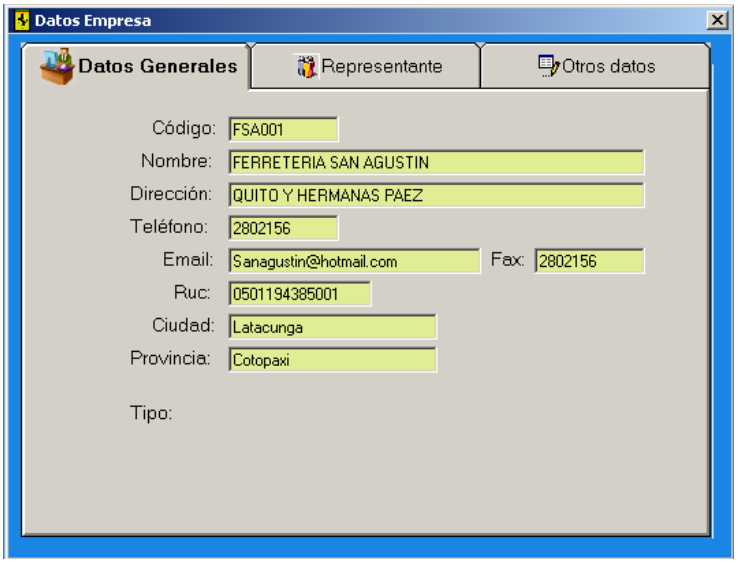


Figura 4.25. Submenú Datos Empresa.

El submenú base datos se ilustra en la Figura 4.26.

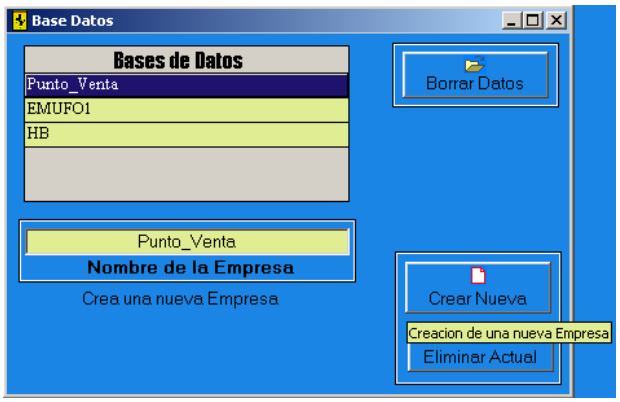


Figura 4.26. Submenú Base Datos.

El menú cartera y su respectivo submenú, se ilustra en la Figura 4.27.

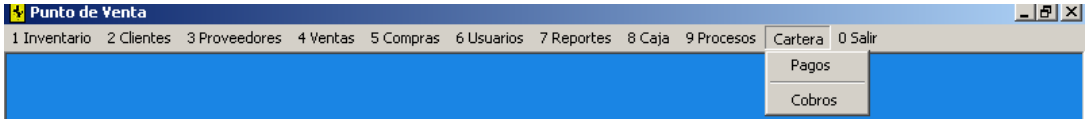


Figura 4.27. Menú Cartera y Submenú.

El submenú pagos se ilustra en la Figura 4.28.

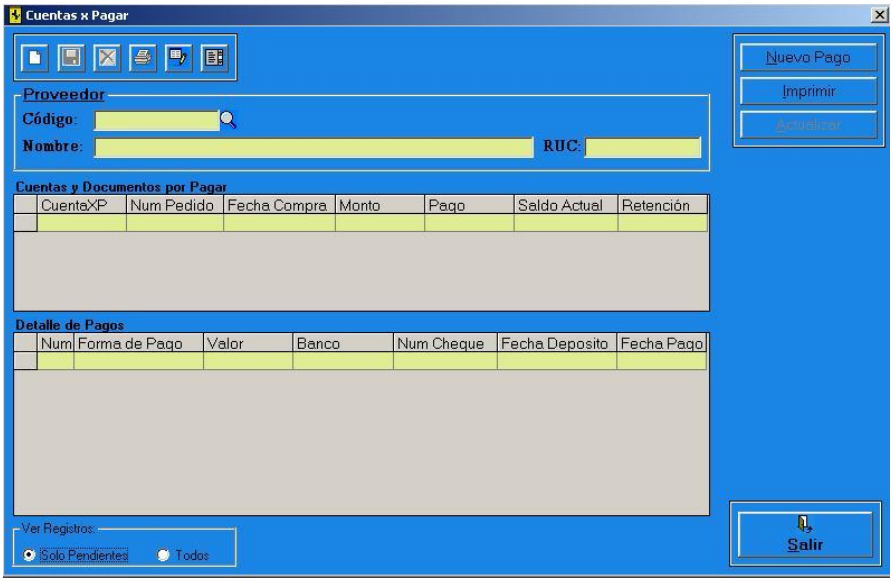


Figura 4.28. Submenú Pagos.

El submenú cobros se ilustra en la Figura 4.29.

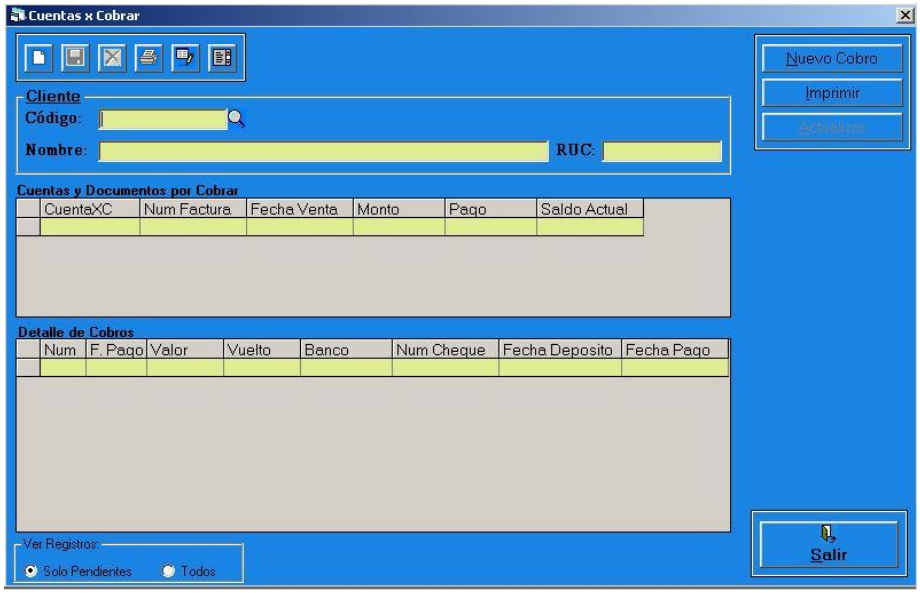


Figura 4.29. Submenú Cobros.

4.3. Diseño de Salidas

En el diseño de salidas se presenta los Reportes y Consultas que nos emite el Sistema, todos los informes según de informe tienen el mismo formato, a continuación se presenta el formato de los reportes que se emiten.

Formato de Listados, se ilustra en la Figura 4.30.

FERRETERIA SAN AGUSTIN FSA

Dirección: Calle Quito y Hermanas Páez
Teléfono: (03) 2812-797
Latacunga - Ecuador

FERRETERIA
"SAN AGUSTIN"
RUC: 0501194385001

Listado de Items
Fecha actual: 18/03/2005

Código	Descripción	Cantidad	Precio U	Bod 1	Bod 2	Bod 3	Bod 4
FSA00001	ABRAZADERA HIDRO 1/2	101	0,40	97	4	0	0
FSA00002	ABRAZADERA HIDRO 3/4	38	0,33	38	0	0	0
FSA00003	ABRAZADERAS IDEAL 1	19	0,33	19	0	0	0
FSA00004	ABRAZADERAS IDEAL 1	25	0,33	25	0	0	0
FSA00005	ABRAZADERAS IDEAL 1/2	5	0,33	5	0	0	0

Figura 4.30. Reporte Listado de Ítems.

Formato de Reportes, se ilustra en la Figura 4.31.

FERRETERIA SAN AGUSTIN FSA

Dirección: Calle Quito y Hermanas Páez
Teléfono: (03) 2812-797
Latacunga - Ecuador

FERRETERIA
"SAN AGUSTIN"
RUC: 0501194385001

Reporte
Items - Proveedor
Fecha actual: 18/03/2005

Código: P002 **Nombre:** MARRIOTT S.A **Última Compra:** 02/11/2005

Código	Descripción	Cantidad	Costo Actual
FSA00001	ABRAZADERA HIDRO 1/2	101	0,30
FSA00002	ABRAZADERA HIDRO 3/4	38	0,40

Figura 4.31. Reporte Ítems-Proveedor.

Formato de Facturas, se ilustra en la Figura 4.32.

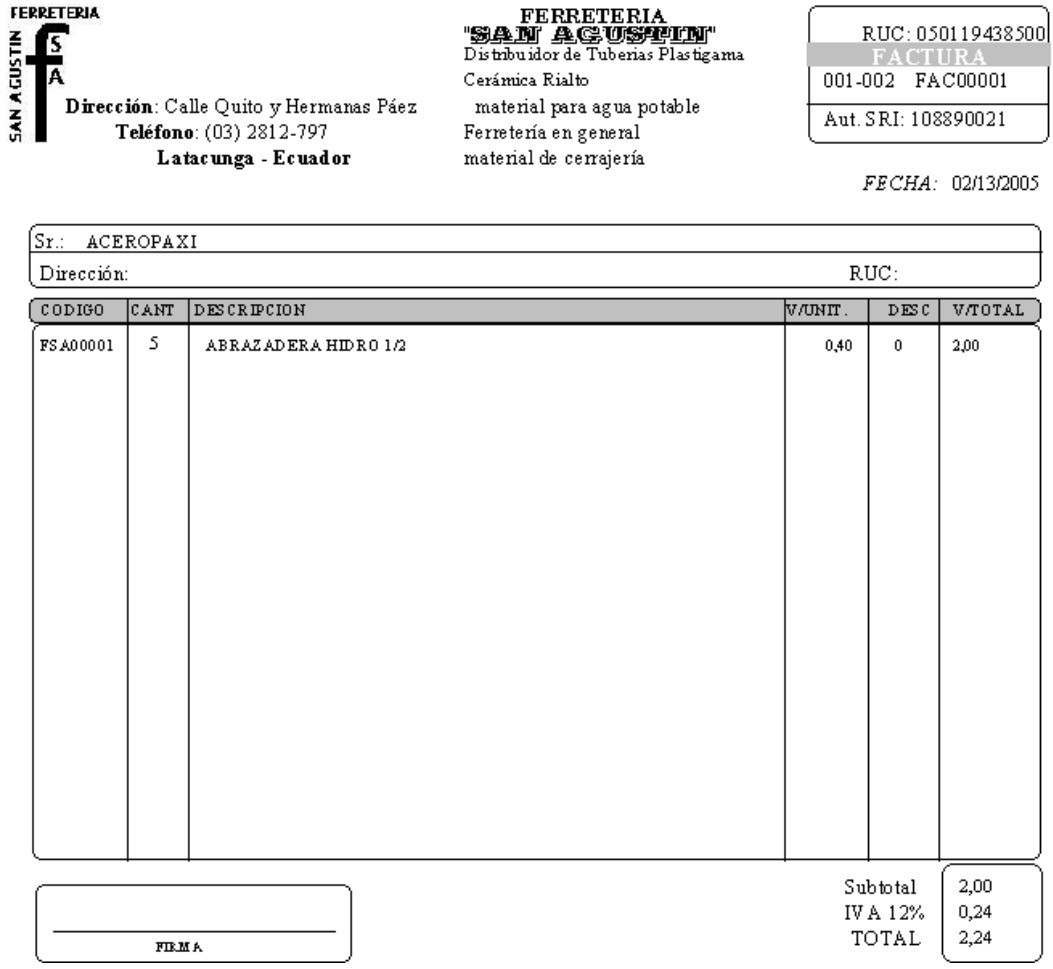


Figura 5.32. Reporte Factura Emitida.

Conclusiones y Recomendaciones

Conclusiones

- Se desarrollo un Sistema Cliente/Servidor aplicado a puntos de venta a través de Sockets. Caso práctico Ferretería San Agustín, la cual esta haciendo uso del presente Sistema.

- Con el funcionamiento de este Sistema se mantendrá un inventario en línea e información actualizada en cada punto de venta.

- Se ha logrado un Sistema de fácil uso, capaz de ser utilizado por usuarios con poca experiencia en Computación.

- Los sockets nos permiten establecer la comunicación entre dos puntos de contacto los cuales pueden estar ejecutando distintas aplicaciones, a través de las cuales estas se comunican.

- El control Winsock de Visual Basic 6.0, permite un acceso sencillo a los servicios de red TCP (Protocolo de control de transferencia) el cual permite crear y mantener una conexión con un equipo remoto y UDP (Protocolo de datagramas de usuarios) que es un protocolo sin conexiones. A diferencia de las operaciones de TCP, los equipos no establecen una conexión.

- Para recibir y enviar información en una red utilizando Sockets en red, se debe especificar las direcciones IP.

Recomendaciones

- Para evitar pérdida de información, actualización en las ventas, pérdida de datos y daños en los equipos se recomienda que los equipos estén conectados a un UPS, ya que de forma imprevista se presenta de vez en cuando apagones.

- Se recomienda incrementar la memoria Ram de los equipos de trabajo a 128 MB, para un mayor aprovechamiento de los recursos.

- Las contraseñas de los usuarios del Sistema deben ser personales y no deben conocer los demás usuarios.

- Para la conexión en red entre computadores se recomienda la utilización de un Swicht Hub, el cual realiza una mejor distribución de la información.

- Se recomienda que la persona encargada de administrar el servidor tenga conocimientos informáticos y una amplia experiencia en el uso de base de datos, para evitar ciertos inconvenientes que suelen suceder de manera imprevista, en el uso de Sistemas.

- Se recomienda en una aplicación Cliente/Servidor si su comunicación es a través de sockets, que en la parte del Servidor el socket este siempre en escucha, y el Cliente cierre el socket cuando este haya recibido información o haya sido atendido por el servidor, para que así el resto de clientes tengan acceso a la comunicación con el Servidor, ya que este da atención a un cliente y luego si tiene petición da atención a otro cliente.

Bibliografía

Libros

- BIRNIOS Baltazar y Mariano, Creación de Aplicaciones Multimedia con Visual Basic, editorial MP Ediciones, primera edición, Buenos Aires, 1998.

- BIRNIOS Baltazar y Mariano, Microsoft Visual Basic Manual de Referencia, editorial MP Ediciones, primera edición, Buenos Aires, 1999.

- BRANCKEK Bob, Microsoft Sql Server 6.5, editorial Prentice Hall, España, 1996.

- J. BENAVIDES, Sql para Usuarios y Programadores, editorial Paraninfo, España, 1991.

- MICROSOFT Corporation, Microsoft Visual Basic 6.0, Manual del Programador, Mc Graw Hill

- PRESSMAN Roger, Ingeniería del Software, editorial McGraw Hill, España, 1994.

- PRESSMAN Roger, Ingeniería del Software, Un enfoque practico, editorial Mc Graw Hill, cuarta edición.

- REYES PONCE Agustín, Administración de Empresas, editorial Linusa, México, 1990.

- SEFFREYP Mc Manus, Base de datos con Visual Basic 6.0, editorial Prentice Hall.

- SENN James, Análisis y Diseño de Sistemas de Información, editorial McGraw Hill, México, 1990.

- SOUKUP Ron, Afondo Microsoft Sql Server, editorial McGraw Hill, 1997.

- STOLTZ Kevin, Todo acerca de las Redes de Computadoras, editorial Prentice Hall, México, 1994

Internet

http://msdn.microsoft.com/vbasic
http://www.vb-helper.com/
http://www.vbexplorer.com/
http://www.cgvb.com/
http://www.vb-world.net/
http://www.planet-source-code.com/vb
http://www.programacion.net/
http://www.programmersheaven.com/
http://www.dominiopublico.com/intranets/cliente_servidor.php
http://www.arrakis.es/~dmrq/beej/index.html
http://www.lofware.com/
http://www.pisuerga.inf.ubru.es/lsi/Docencia/TFC/ITIG/icruzadn/Memoria/index.
http://www.monografias.com

Glosario de Términos

ASCII

Código estándar empleado para convertir textos en datos binarios legibles por cualquier ordenador.

BPS

Unidad de velocidad de transmisión de datos (bits por segundo). Es importante en un módem y se refiere al número máximo de bits o unidades de información que pueden transmitirse cada segundo de tiempo por una línea de teléfono convencional en condiciones óptimas.

Cliente/Servidor

Concepto que, en informática de redes, sirve para clasificar los ordenadores: los servidores almacenan información y la entregan a los clientes, que la solicitan.

Concentrador

Punto de conexión central para un conjunto de dispositivos de una red configurada en estrella. Actúa a modo de agente de tráfico, dirigiendo la transmisión de los datos entre dichos dispositivos. El número de nodos conectados a un concentrador está limitado por los puertos disponibles de éste pero, si se necesita que la red soporte un número mayor de nodos, se pueden conectar varios concentradores.

Dirección IP

Ver TCP/IP.

Encriptación

Proceso de conversión de un texto en otro, cifrado. Se emplea para ello un código que impide la lectura de información reservada a cualquiera que no sea el destinatario poseedor de un software y una clave que permite descifrarla.

Hub

Dispositivo de interconexión, punto central de conexión para nodos de red que están dispuestos de acuerdo a una topología física de estrella.

Internet

Red mundial de ordenadores, tanto ordenadores personales como superordenadores, que emplean el protocolo TCP/IP para comunicarse. Ofrece una gran cantidad de servicios a todo el que esté conectado a ella.

Intranet

Red local que utiliza, total o parcialmente, las tecnologías de la Internet.

ISDN

Siglas de Integrated Services Digital Network, Sistema de transmisión de datos que en España recibe el nombre de RDSI.

Java

Lenguaje de programación. En la Web es utilizado para incluir pequeños programas en las páginas.

LAN

Siglas de Local Area Network, se trata de una red local, esto es, instalada en una misma sala o edificio.

Módem

Dispositivo (Modulador De modulador) utilizado para transmitir datos a través de conexiones telefónicas estándares. Generalmente, se conecta a un puerto serie de un ordenador y a una clavija telefónica RJ-11. Los datos son aceptados por el puerto serie del ordenador en forma original y son convertidos por el módem a una serie de tonos analógicos que son transmitidos por la línea telefónica en un proceso de conversión digital-analógica.

Protocolos

Normativas estándar para las telecomunicaciones en general y la comunicación entre ordenadores en particular. El software de los ordenadores en red tiene que diseñarse para cumplir esta normativa. Ciertos protocolos se usan en las transferencias realizadas sobre LANs físicas (ApleTalk y PPP, entre otros) y otros como FTP y HTTP se "montan" sobre los primeros para completar la distribución de contenidos multimedia, correo electrónico, archivos de datos, etc.

- **FTP**

  Siglas de File Transfer Protocol, o protocolo de transferencia de ficheros, utilizado para transferir éstos en redes que utilizan TCP/IP. Muy usado, por tanto, en Internet.

- **PPP**

  Protocolo punto a punto (Point to Point Protocol), fue desarrollado en su origen para la comunicación entre encaminadores. Ahora se utiliza también en dispositivos instalados en RDSI.

- **TCP/IP**

  Siglas tomadas de Transfer Control Protocol/Internet Protocol, es el protocolo en el que se basa todo el tráfico en Internet. Cualquier ordenador en Internet debe tener una única dirección IP. Ésta consta de

un conjunto de cuatro cifras de la forma xxx.xxx.xxx.xxx donde xxx puede ser un número comprendido entre 0 y 255. Los ordenadores de Internet utilizan esta dirección para llamar a otros, de manera similar a como lo hacen los teléfonos.

- **HTTP**
  Siglas de Hyper Text Markup Language, es el protocolo de transferencia de transporte de hipertexto, usado en la World Wide Web. Se utiliza para solicitar una página de un servidor Web y para que el servidor vuelva a transferir información multimedia. No es un lenguaje propiamente dicho, sino texto ASCII que puede ser interpretado por los diferentes navegadores.

- **SMTP**
  El protocolo simple de transporte de correo (Simple Mail transfer Protocol) es el usado para transferir correo sobre redes que utilizan TCP/IP. Es el protocolo del correo electrónico (EMail) de Internet.

Proxy
Programa servidor que sirve para que el ordenador cliente acceda a datos de manera mucho más rápida y eficaz.

RDSI
Siglas de Red Digital de Servicios Integrados, Sistema de transmisión de datos de gran éxito en España por la gran velocidad de transmisión de datos que permite (alrededor de los 64.000 bps).

Servidor
Todo ordenador que permite a otro conectarse a él mediante un programa cliente, para compartir información y recursos.

URL

Siglas de Uniform Resource Locators, direcciones para localizar diferentes recursos dentro de Internet (sitios Web, sitios FTP, sitios Gopher, etc.).

WAN

Derivada de Wide Area Network, la expresión alude a una red de área ancha o a un conjunto de redes locales conectadas entre sí.