ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"SISTEMA VENTA DE ELECTRODOMESTICOS ORIENTADO A INTERNET UTILIZANDO LA TECNOLOGIA CORBA"

Trabajo de Graduación

Previo a la Obtención del Título de:

INGENIERO EN COMPUTACION

Presentado por:

Eleanor Alexandra Varela Tapia

GUAYAQUIL - ECUADOR

Año 1.999

DEDICATORIA

Dedico la culminación de la tesis a Dios, a *mis* padres, a GALI y a todos aquellos que desearon de forma directa ó indirecta que salga adelante **para** la obtención de **mi** título.

AGRADECIMIENTO

Doy gracias al Padre Celestial por ayudarme alcanzar unas de mis metas más deseadas, por escuchar mis ruegos ya que en algunos momentos pens4 que no iba poder culminar la elaboración del proyecto.

Y a GALI una persona muy importante en mi vida, ya que por su continuo apoyo incondicional estuvo siempre presto a darme una palabra de aliento, fuerzas y energia para poder salir adelante.

TRIBUNAL DE GRADO

ING. CARLOS MONSALVE
Presidente del Tribunal

ING. CARLOS VALERQ
Director del Tópico

ING. GUIDO CAICEDO
Miembro del Tribunal

ESTRADA
Miembro del Tribunal

DECLARACION EXPRESA

" La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, corresponden exclusivamente a su autor, y **el** patrimonio intelectual de la Tesis de Grado corresponderá a la Escuela Superior Politécnica del Litoral ··

ELEANOR ALEXANDRA **VARELA** TAPIA

Resurnen

Este documento recopila toda la información acerca del análisis, diseño e implementación para el proyecto de graduación, utilizando la Tecnología **CORBA**.

El tema asignado para desarrollar el proyecto, fué un Sistema de Venta de Electrodomésticos. Una vez obtenido el tema, se realizó un análisis detallado sobre cada uno de los requerimientos basicos y opcionales dados por el Director del Tópico de Graduación, el Profesor el Ing. Carlos Valero.

El análisis del Sistema *se* lo ha realizado tomando en consideración los requerimientos del profesor y adicionalmente de otros Sistemas orientados al mismo tema. De esta manera se obtuvo un mejor enfoque sobre los requerimientos y procesos **para** la elaboración del mismo.

El Sistema desarrollado *es* una aplicación 3 tier, es decir que esta estructurado en **3** procesos o fases:

- Cliente Applet
- Servidor : Compuesto por un WebServer y un Servidor Corba
- Servidor de Rase de Datos

El Cliente Applet es un proceso que se ejecuta dentro de un browser, cuando el usuario realice algún requerimiento al Servidor para ejecutar una transaccihn, este llegará inicialmente al WebServer para luego comunicarse con el Servidor Corba.

El Servidor Corba es un proceso que ejecutará cada una de las transacciones requeridas por el cliente, extrae la información necesaria de la base de datos y retorna la respuesta.

El Servidor de Base de Datos es un proceso que permite crear, administrar, actualizar y mantener los datos.

Adicionalmente el Sistema utiliza la tecnología Internet, brindando facilidades al cliente para que pueda realizar sus compras desde su hogar u otro sitio, a través de vendedores.

Espero que la lectura de este documento y la aplicación del Sistema, sea del máximo provecho para los estudiantes de Computación, ya que he volcado todo mi máximo esfuerzo posible para que así sea.

Indice General

Introducción

El presente documento describe el diseño e implementación de un Sistema de Venta de Electrodomésticos Cliente/Servidor, usando la Tecnología CORBA.

El Sistema es una aplicación que está enfocada a pequeñas ó medianas empresas que se dedican a vender electrodomésticos. Está ideado para ser utilizado a través de Internet, de manera que el cliente pueda realizar la compra de algún electrodomkstico desde su hogar u otro sitio por intermedio de vendedores, sin necesidad de que se acerquc al almacén.

Como primer paso se menciona los requerimientos del Sistema, para describir los objetivos y alcance del mismo.

Posteriormente se explica en qué consistc la arquitectura Cliente/Servidor. Luego se define el concepto de CORBA, cuáles son sus componentes, sus formas de invocación (Estática, Dinámica y Servicio de Nombramiento) entre otros conceptos.

Luego se hace un análisis del Sistema, utilizando la metodología Orientada a Objetos.

Como el Sistema está ideado a través de Internet, *se* describe los mecanismos y protocolos de comunicación usados.

Se explica también la estructura de la base de datos, la misma que ha sido implementada usando el Sistema SQL Server.

Finalmente se describe las conclusiones, entre las que se mencionan: Los objetos pueden encontrarse distribuidos en la red, éstos pueden comunicarse sin necesidad de conocer su localización dentro de la red; NO importa en que lenguaje dc programación estén implementados ni la plataforma en quc operan.

Capítulo 1

Generalidades

1.1 Descripción del Sistema

El Sistema es una aplicación Cliente/Servidor orientado a Internet, utilizando la Tecnología CORBA.

Está formado por 3 partes principales:

1. Proceso Cliente
2. Proceso Servidor
3. Administrador de Base de datos

Proceso Cliente: Como es una aplicación orientada a Internet, este es un applet que se ejecuta dentro de un browser, cuando el usuario realiza un requerimiento, el cliente se comunica con el WebServer para obtener la página html que contiene el menú principal, comunicándose así con el Servidor Corba, posteriormente el cliente envía el requerimiento y queda esperando hasta que el Servidor le retorne la respuesta.

Proceso Servidor: Está formado por 2 subprocesos que son: El WebServer y el Servidor Corba. El WebServer es un intermediador que ayudará a establecer la

comunicación entre el Cliente Applet con el Servidor Corba a través del middleware ORB, al principio cuando reciba el primer requerimiento del Cliente Applet, éste le enviará la página html del menú principal y cuando reciba los requerimientos para ejecutar cada una de las diferentes transacciones el WebServer le enviará lo siguiente: La nueva página html, el applet, el Stub y el ORB. El Servidor Corba establece la conexión con el Servidor de Base de Datos a través de los drivers JDBC/ODBC, ejecuta la scntencia SQL de acuerdo a la transacción requerida y envía la respuesta al Cliente Rpplet.

El Administrador de Base de Datos: Esta aplicación ha sido usada para implementar, administrar y llenar de información la base de datos.

1.2 Objetivos

- Implementación de un Sistema 3-tier Cliente/Servidor orientado a Internet, utilizando la Tecnología CORBA.
- Aplicar los conocimientos aprendidos en el tópico de graduación: La arquitectura Cliente/Servidor, la Tecnología CORBA y los Administradores de Rase de Datos.
- Accesar a la Base de Datos por medio de los drivers JDBC/ODBC.
- Enfocar el uso del Sistema Venta de Electrodomésticos.

1.3 Arquitectura Cliente/Servidor

Es una Arquitectura usada para diseñar software de aplicación, dando como resultado la subdivisión de un sistema de información en 2 procesos. Estos son:

- Proceso Servidor
- Proceso Cliente

Proceso Servidor: Es un proceso especializado en proveer servicios, tales como datos, información, procesamiento, etc.; a un conjunto de procesos clientes.

Proceso Cliente: Es un proceso que pide servicios a un proceso servidor, a través de requerimientos.

Para que ambos procesos cliente y servidor puedan comunicarse para intercambiar datos, necesitan de un mecanismo de comunicacicin, este se lo conoce con el nombre de **"Middleware".** El mismo que se relaciona con la red y los protocolos de comunicación.

1.3.1 Componentes de Cliente/Servidor

Lado del Cliente: Existen **4** partes fundamentales:

1. Interfaces
2. Procesos
3. Herramientas
4. Acceso a Datos

Interfaces: Corresponden al GUI, es decir a las interfaces gráficas que serán usadas por el usuario para manejar el sistema

Procesos: Corresponden propiamente a la lógica del cliente, es decir: Envío de requerimientos, validación de las peticiones y mostrar los resultados en la pantalla.

Existen 2 tipos básicos **de** clientes: Clientes Independientes y Clientes basados en Browser.

Los Clientes Independientes.- Se los conoce como Standalone **y** son aquellos **que** son desarrollados con herramientas que generan productos compilados.

Los Clientes basados en Browser.- Son aquellos que pueden ser solo ejecutados dentro de un browser.

Henamientas: Son los productos que permiten tanto el acceso a la Base de Datos, como al desarrollo de aplicaciones. Toda herramienta en el proceso cliente se la denomina *"Front-End"* **y** en el proceso servidor es llamado *"Buck-End':*

Acceso a Datos: Es el middleware (mecanismo de comunicación) para la conexión con la base de datos a través del Servidor.

Lado del Servidor: Existen 2 partes fundamentales:

1. Procesos
2. Acceso a Datos

Procesos: Comprenden la lógica del Servidor, es decir procesar cada uno de los requerimientos generados por los procesos clientes y enviarlos al cliente originario del requerimiento.

Acceso a Datos: Comprenden el acceso a la base de datos, el servidor obtiene los datos necesarios a procesar para luego guardar los resultados en la base de datos.

Lado del Middleware: Es un conjunto de procesos intermedios entre Clientes y Servidores. Está constituido por 3 componentes:

1. Mecanismos de Comunicación
2. Funciones Especiales de NOS
3. Servicios Específicos

Mecanismos de Comunicación: Se refiere a los protocolos de comunicación al nivel de las capas de Sesión, Transporte y Red, los mismos que permiten establecer sesiones o conexiones entre clientes y servidores. Ejemplo: TCP/IP(sockets), NetBEUI, Named Pipes, etc.

Funciones Esueciales de NOS Son funciones de apoyo que extiendcn el alcance y la capacidad de los Sistemas Operativos de las computadoras empleadas entre clientes y servidores. Ejemplo: Funciones para la invocación remota de procedimientos y procesos servidores (RPC), Funciones de Administración para archivos distribuidos, etc.

Servicios Específicos: Incluye todas las otras funciones utilizadas para la resolución de problemas específicos, es decir, son funciones muy ligadas y dependientes principalmente del tipo del servidor con el que se está operando.

Consiste de **4** categorías básicas: Middleware Transaccional, Middleware Groupware, Middleware Database y Middleware de Objetos Distribuidos.

Middleware Transaccional.- Son herramientas utilizadas por Servidores de Transacciones como TP-Monitors, TP-Heavy (DBMS).

Middleware Groupware.- Son herramientas, funciones y protocolos utilizadas por Servidores de Grupos como WorkFlow Tools.

Middleware Database.- Son herramientas utilizadas por Servidores de BD como ODBC, JDBC, ejecución de procesos (stored procedures, triggers, rules).

Middleware Objetos Distribuidos.- Son herramientas, procesos y protocolos requeridos por Objetos Servidores como CORBA, DCOM.

1.4 ¿Qué es CORBA

El término CORBA significa Commom Object Request Broker Architecture, es una nueva Tecnología de Integración para Sistemas Cliente/Servidor orientado a objetos distribuidos.

CORBA fue introducido en al año **1991** por Object Management Group (OMG), esta organización define el IDL (Interface Definition Languaje) y un APT (Application Programming Interfaces) para implementar la interacción dc objetos entre los procesos clientes y servidores en un medio distribuido.

La idea de CORBA es simple. Hacer fácil el uso de objetos distribuidos usando una arquitectura estándar abierta. Con ésta Tecnología, la aplicación no se preocuparía de la ubicación del servidor, protocolos de comunicacibn, plataforma de operación, lenguaje de implementación de sus módulos (objetos), etc.

En un ambiente distribuido, *se* necesita de un *"intermediario"* que ayude a los procesos clientes encontrar procesos servidores y puedan atender sus requerimientos. Este intermediario es el componente principal de CORBA, se lo llama ORB (Object Request Broker). El ORB abarca todo lo referente a la infraestructura de comunicación necesaria para identificar y localizar objetos, maneja la conexión y libera los datos.

En esta arquitectura el cliente requiere servicios de objetos (los cuales podrían llamarse "servidores") a través de una **interface** muy bien definida llamada IDL.

1.4.1 ORB

Definición: El ORB es un proceso que trabaja como middleware para cstablecer la comunicación entre el clicnte y el servidor.

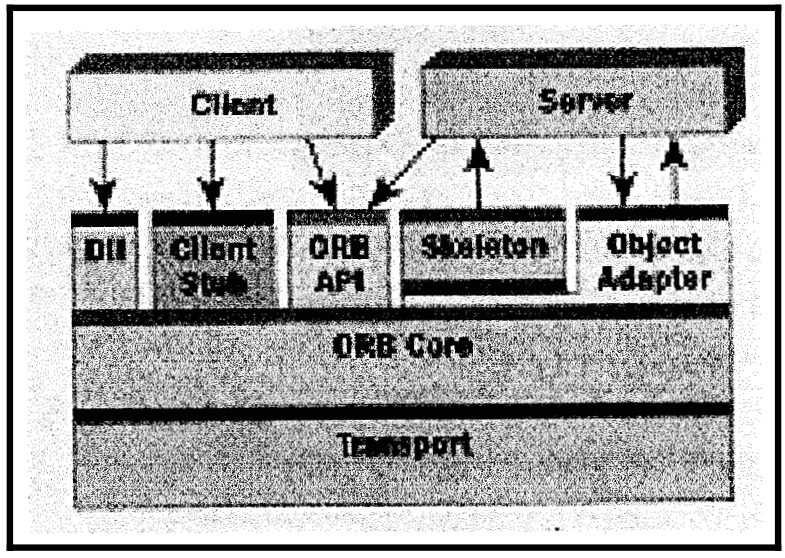


figura 1.1

A través del ORB, el cliente puede invocar un método de manera transparente a un objeto servidor que puede estar en el mismo computador o en la red. El ORB intercepta la llamada generada por el cliente y es el responsable de encontrar el objeto que implemente el requerimiento del cliente, pasa los parámetros, invoca los métodos y retorna los resultados. El cliente no tiene conocimiento donde se encuentra el objeto, en que lenguaje de programación ha sido implementado, ni cual es la plataforma en donde se ejecuta. Todo este trabajo lo hace el ORB, por *eso* en la arquitectura CORBA existe interoperabilidad entre aplicaciones de diferentes máquinas por lo que la característica principal de *CORBR* es operar aplicaciones en un ambiente heterogéneo distribuido.

En síntesis el ORB es simplemente un proceso, cuya función es localiaar los procesos que proveerán los servicios requeridos. Una vez localizados, el cliente puede conectarse con el servidor.

1.4.1.1 Estructura del ORB del Lado del Cliente

El cliente se comunica con el ORE3 para poder mandar un requerimiento al servidor, ya sea a travks dc la interface de invocación dinámica DII ó del IDL stub. El IDL stub hace crecr al cliente que los objetos requeridos se encuentra localmente.

El depósito de interfaces es una base de datos que almacena información de las interfaces definidas.

1.4.1.2 Estructura del ORB del Lado del Servidor

El Objeto Implementado (OI), recibe un requcrimicnto como una llamada ya sea a través de un Static IDL skeleton ó del Dynamic skeleton la cual puede **ser** invocada por un stub IDL ó por un DII del lado del cliente.

El Objeto Adapter (OA), es el medio que permite la comunicación entre el ORB y el OI.

El depósito de implementaciones almacena información que permite al ORB localizar los objetos de control, administrativo, seguridades, etc.

1.4.2 IDL

Es un lenguaje de notación estándar desarrollado por la OMG para definir interfaces. Ofrece una notacibn universal para la especificación de **API,** además soporta interfaces de funciones de biblioteca, asi como objetos distribuidos a lo largo de una red.

La interface, es el medio que ve únicamente el cliente, además es completamente independiente de donde está el objeto y en que lenguaje de programación está implementado. En otras palabras, a través de las interfaces CORHA logra la integración de aplicaciones distribuidas.

1.4.2.1 Sintaxis del IDL

Su sintaxis es muy parecido a C++. Esta estructurado en 3 partes:

1. Atributos
2. Operaciones o métodos
3. Parámetros

Atributos: Son las variables o atributos de los objetos.

Sintaxis: attribute <tipo_atributo> <nombre_atributo> ;

Operaciones *6* Métodos: Son las funciones o métodos de los objetos.

Sintaxis: <tipo_retorno_método><nombre_método>
(<dirección_parámetro><tipo_parámetro>
<nombre_parámetro>, ...)

Parámetros: Son los argumentos de los métodos de los objetos.

Sintaxis: In <tipo_parámetro> <nombre_parámetro>

Out <tipo_parámetro> <nombre_parámetro>

InOut <tipo_parámetro> <nombre_parámetro>

Donde:

In .- Si el parhetro es pasado del cliente al objcto llamado.

Out .- Si el parhetro es pasado desde el objeto llamado al cliente.

InOut .- Si el parhetro es pasado en ambas direcciones.

Ejemplo:

```
//Módulo IDL
module nombre_idl{
  interface nombre_inteface    {
      attribute tipo nombre_atributo;
      tipo nombre_método (in argumentol, out argumento2,
                                inout argumento3);
  }
}
```

1.4.2.2 Mapeando IDL a Java

Por medio de una interface IDL se obtiene un paquete Java, el mismo que contiene interfaces y clases que serán utiliaadas por el cliente con el servidor para poder cornunicarse por medio del ORB.

Estas clases e interfaces generadas son:

- module nombre_idl._st_interface
- module nombre_idl.interfaceHelper
- module nombre_idl.interfaceHolder
- module nombre_idl._interfaceImplBase
- module nombre_idl.interface
- module nombre_idl._example_interface

module nombre_idl nombre_idl._st_interface :Clase de Java que implementa el stub del lado del cliente.

modulo.interfaceHelper : Clase de Java que provee una serie de funciones para que el cliente por medio de la interface encuentre el objeto en el ORB.

modulo.interfaceHolder : Clase de Java que mantiene una instancia a la interface, es usada por los clientes y servidores para pasar objetos del tipo de la interface como haya tantos parámetros de in y out dentro de los métodos de invocación.

modulo._interfaceImplBase : Clase de Java que implementa el skeleton del lado del servidor.

modulo.interface : Es una interface en Java. Este mapea la interface IDL a un paquete Java.

modulo._example_interface : Clase de ejemplo para desarrollar la lógica del objeto implementado

1.4.3 Tipos de Invocacion

Existen *3* tipos de invocación que el cliente utiliza para localizar al objeto implementado del servidor por medio del ORB, los mismos que son:

1. Invocacicin Estática
2. Invocacicin Dinámica
3. Invocación por medio de un Servicio de Nombramiento

1.4.3.1 Invocacion Estática

El clientc ve únicamente la interface del Servidor, por lo que el cliente realiza una referencia a la interface para localizar al objeto implementado dcl scrvidor que se encuentra en el ORB. Si el objeto servidor es remoto, la referencia apunta a una función STUB y utiliza la estructura ORB **para** invocar al objeto servidor. El código

STUB usa al ORB para identificar la máquina que corre el objeto servidor y se conecta a ella. Luego, el STUB envía la referencia y los parámetros al código skeleton (lado del servidor). Este transforma la llamada y parámetros en una implementación requerida. Cualquier resultado es retornado en la misma vía, ver figura 1.2.

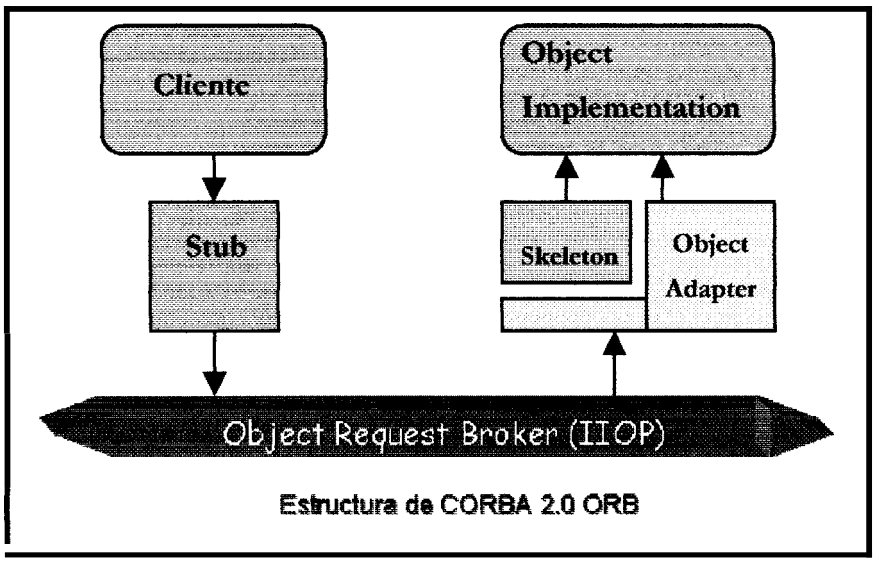


figura 1.2

La característica principal de la invocación estatica, es que la interface IDL se encuentra previamente precompilada y mapeada a un paquete Java.

Caracteristicas:

- Fácil programación, ya que su sintaxis es muy parecida a C++.
- Provee un ambiente robusto para reconocer errores de compilación, ya que el IDL se encuentra previamente precompilado.

- Tiene un conjunto completo de **API,** invocado por el stub para realizar requerimientos al ORB.
- Rápido entendimiento de la interface IDL, no se necesita documentación.
- Existe un stub por cada interface que el cliente usa.

Pasos para realizar la invocación estática:

1. Definir la interface del servidor usando el lenguaje IDL.
2. Colocar las definiciones **de** las interfaces en **el** repositorio de interfaces.
3. Correr el archivo IDL con el precompilador propio **de** CORBA.
4. Construir los objetos implementados.
5. Compilar los objetos implementados.
6. Registrar los objetos en el repositorio de implementación.
7. Instanciar los objetos en el servidor.
8. Implementar el código del cliente.
9. Compilar el código del cliente.

1.4.3.2 Invocación Dinamica

La invocación dinámica permite que el cliente tome d a r e s de objetos en una red, sin requerir **un** stub precompilado por cada interface del servidor.

Catacteristicas:

- Permite a los clientes tomar un objeto en tiempo de ejecucion y dinámicamente invocar sus métodos.
- Los clientes pueden invocar un método de cualquier objeto sin requerir stub precompilados.
- Los clientes pueden conocer los objetos del servidor en tiempo de ejecución y saber cómo invocar sus métodos.
- Provee un ambiente muy dinámico que permite a los sistemas ser flexibles y extensibles.

Pasos para realizar la invocación dinámica:

1. Se obtiene una referencia del objeto.
2. Usar la referencia del objeto para recuperar la interface del objeto y dinámicamente construir el requerimiento.
3. En el requerimiento se debe especificar el método a ejecutar y sus parámetros, esta información se obtiene de un repositorio de interface.
4. Envíar el requerimiento.

I.4.3.3 Servicio de Nombramiento

Este servicio le permite asociar uno o más nombres lógicos a la implementación de un objeto y guardar esos nombres en un servicio de nombramiento.

El servicio de nombramiento se lo puede organizar en un árbol como estructura de objetos que pueden ser atravesados para localizar un nombre en particular.

La asociacicin del nombre al objeto es llamada "name binding". 'I'odo objeto tiene referencia única.

Capitulo 2

Análisis y Diseño

2.1 Requerimiento

Los requerimientos analiaados para construir el Sistema Venta de Electrodomésticos, son:

Venta de Artículo.- Esta opción debe especificar el código del articulo. Los códigos podrían ya estar creados en la base. Si se vende más de un articulo, para cada artículo nuevo se debe verificar tanto su código como su existencia en el inventario. La transaccihn debe retornar el respectivo precio del artículo. Una vea ingresados todos los articulos, la transacción hari un solo envío final. Cada venta generará una factura que tendrá un detalle de los articulos vendidos en esa transacción.

Retorno de Artículo.- Esta opción permite volver a constar en el inventario. El detalle de la respectiva factura también deberá ser actualizado.

Consulta/Emisión de Factura.- Esta opción debe mostrar en pantalla los datos de una venta (articulos, cantidades y precios).

Ingreso de Artículos.- Esta opción debe permitir incrementar el inventario. Debe especificarse el código del artículo ingresado, la respectiva cantidad y el precio unitario **de** venta.

Creación de Nuevos Artículos.- Esta opción debe permitir crear nuevos códigos de artículos, cuyo inventario sería incrementado con la respectiva transaccibn.

Eliminación de Artículos.- Esta opción debe permitir eliminar artículos para darlos de baja.

Ajuste de Inventario.- Esta opción debe permitir realizar un ajuste del inventario para disminuir articulos ya sea por robo ó daño.

Precio de Venta.- Esta opción debe permitir cambiar el precio de venta de los articulos.

2.2 ¿Por que Orientado a CORBA?

Las razones por la cual el proyecto del tópico de graduación fue desarrollado usando la arquitectura CORBA, son:

- Implementar un Sistema Cliente/Servidor Orientado a Objetos Distribuidos.
- CORBA define un middleware que tiene el potencial suficiente para incluir en él cualquier otra forma existente de middleware cliente/servidor. Es decir que la tecnología de objetos distribuidos de CORBA permite conjuntar complejos sistemas de información de cliente/servidor con sólo ensamblar y extender componentes.
- CORBA fue diseñado para permitir que cornponentes inteligentes se descubran entre sí e interoperen en un middleware de objetos (ORB).
- CORHA permite crear un objeto ordinario para después hacerlo transaccional, seguro, bloqueable y persistente mediante el recurso destinatario de herencias múltiples de los servicios apropiados.
- Los objetos de CORBA son entidades inteligentes que pueden vivir en cualquier punto de una red.

2.3 Modelo 3-tier

La estructura del sisterna está conformada por *3* capas, por eso corresponde a un modelo 3-tier. Estas capas son:

1. El Cliente Applet
2. El Servidor conformado por: WebServer y el Servidor Corba
3. Servidor de Base de Datos

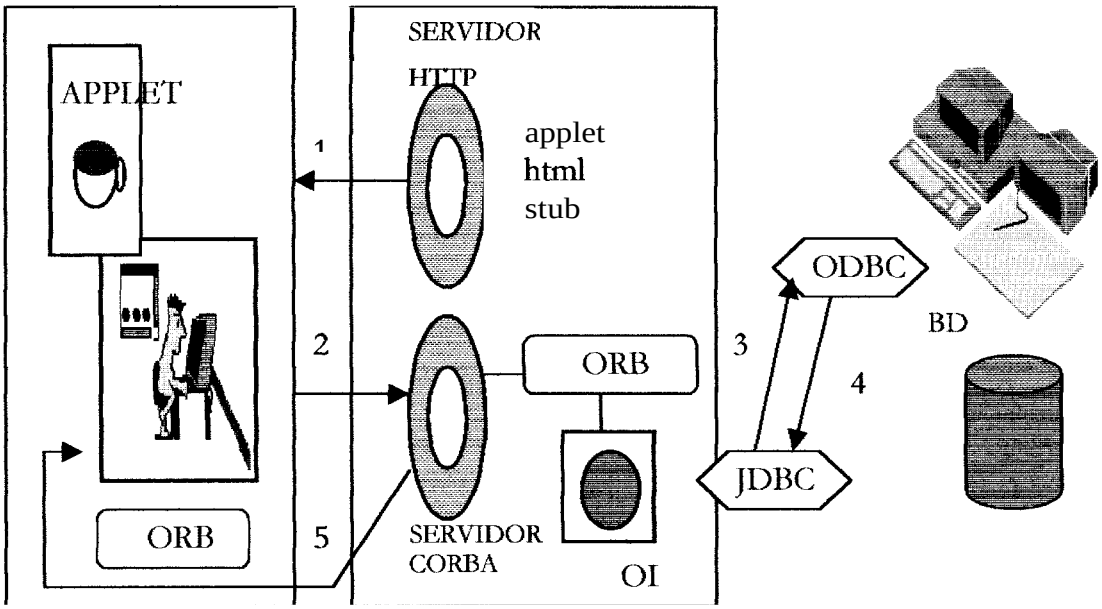


figura 2.1

El usuario realiza un requerimiento, este es capturado por el cliente applet, el cual envía el requerimiento via HTTP al WebServer, este recoge el requerimiento y le envía la página del menú inicial al cliente, entonces el usuario podrá ingresar al sistema. Luego el usuario ejecutará cada una de las diferentes transacciones, el cliente las enviará y serán receptadas por el WebServer quien le enviará según el requerimiento la página html, el applet, las clases de CORBA (stub e interface) y el ORB. Cuando el usuario realice un requerimiento en particular, el cliente invocará al objeto correspondiente que está en el ORB, el cliente creerá que el objeto se encuentra localmente en su estación, pero en realidad el cliente a través de la interface

podrá invocar los métodos que necesite, ya que la interface recoge la llamada a través del stub del cliente, la pasa al ORB para encontrar el objeto servidor pasando los parámetros, los mismos que son receptados por el skeleton del servidor para yue lleguen al objeto implementado, finalmente la respuesta es retornado por la misma via sin que el cliente se halla enterado donde estaban los objetos servidores.

Los objetos implementados, previamente fueron exportados al ORB por el Servidor Corba quien a su vez cada objeto implementado realiza una instancia a un objeto de acceso a la base de datos para comunicarse con el Servidor de Base de Datos y extraer los datos como resultados de las respectivas transacciones ó sentencias SQL: SELECT, INSERT, UPDATE, DELETE.

2.4 Análisis

Para implementar un sistema orientado a objetos distribuidos usando la arquitectura CORBA, se debe realizar un análisis exhaustivo para poder identificarlos objetos que interactuarán en el sistema. Uno de los métodos usados para poder identificar los posibles objetos es a través de los casos de usos.

Un caso de uso es una tabla donde se describe paso a paso cada unas de las metas principales que se desea alcanzar para elaborar el sistema, detallando quienes son los actores primarios (usuarios), actores secundarios (entidad ó sistema externo

para obtener el éxito de la meta) y el escenario (secuencia de interacciones que se dan bajo ciertas condiciones para alcanzar una meta).

Sobre la base de los requerimientos planteados para desarrollar el Sistema Venta de Electrodomésticos en la sección 2.1, se identificaron las siguientes metas:

- Venta de un Artículo
- Consulta y Emisión de Factura
- Actualización del Inventario (Devolución, Incremento y Ajustc)
- Manejo de Articulos (Crear, Eliminar y Consultar)
- Registrar Precios de Venta del Artículo
- Manejo de Clientes (Crear, Eliminar y Consultar)

2.4.1 Casos de Usos

Venta de un Artículo

Caso de Uso	Alcance
Meta	Ingresar factura del sistema
Alcance	Sistema
Nivel	Primario
PreCondiciones	Códigos de los articulos deben existir
PostCondiciones de Exito	Incrementando # facturas existentes
PostCondiciones de Fallo	N° de factura asignado no se registra
Actores Primarios	Vendedor - Almacén
Actores Secundarios	Tabla de artículos (BD)
Trigger	Petición de venta del artículo y generar una factura

Descripción	Pasos	Acción
	1	Se hace requerimiento de ingreso de factura
	2	Sistema coloca siguiente # de factura disponible
	3	Se ingresan datos de cabecera de factura
	4	Se ingresan códigos de articulos
	5	Se trae a la pantalla datos del artículo
	6	Se ingresan cantidades de articulos
	7	Se actualiza el inventario
	8	Se calcula totales de factura

Extensiones	Pasos	Acción
	1	Más de un vendedor realice ingreso de factura

Consulta y Emisión de Factura

Caso de Uso	Alcance
Meta	Consultar y/o pago **de** factura
Alcance	Sistema
Nivel	Primario
PreCondiciones	Factura ya existe
PostCondiciones de Exito	Consultar factura y registrar el pago
PostCondiciones de Fallo	# factura buscada no existe
Actores Primarios	Vendedor – Almacén
Actores Secundarios	Tabla de Factura (BD)
Trigger	Petición de requerimiento de consulta y **pago** de factura

Descripción	Pasos	Acción
	1	Se hace requerimiento de consulta de factura
	2	Sistema captura # de factura para realizar búsqueda
	3	Factura mostrada en pantalla
	4	Registro de pago de la factura en la tabla ventas

Extensiones	Pasos	Acción
	1	Si # factura no existe, mostrar mcnsaje en pantalla

Actualización del Inventario

Caso de Uso	Alcance
Meta	Modificar la cantidad **de** artículos de stock
Alcance	Sistema
Nivel	Primario
PreCondiciones	Articulo ya existe
PostCondiciones de Exito	Si es una compra, la cantidad es > 0 Si **es** una venta ó ajuste, la cantidad es > 0 y < = a la cantidad existente en stock. Si es devolución, la cantidad es > 0 ó < = a la cantidad vendtda.
PostCondiciones de Fallo	Articulo no existe.
Actores Primarios	Vendedor – Almacén
Actores Secundarios	Tabla de Articulos, Factura y Ventas (BD)
Trigger	Petición para aumentar o disminuir el stock

Descripción	Pasos	Acción
	1	Se hace requerimiento de actualizar el inventario ya sea por compra, ajuste ó devolución.
	2	Sistema captura código de artículo y cantidad.
	3	Verificación de la cantidad según el tipo **de** actualización del inventario.
	4	Actualización del inventario.
	5	Muestra los resultados en pantalla

Extensiones	Pasos	Acción
	1	Si código de artículo no existe, mostrar mensaje en pantalla

Manejo de Articulos

Caso de Uso	Alcance
Meta	Crear, eliminar y consultar articulos
Alcance	Sistema
Nivel	Primario
PreCondiciones	1.- Para crear un nuevo articulo, este no debe de existir. 2.- Se eliminan sólo los articulos si fueron dados de baja. 3.- Se consulta sólo los artículos activos y con códigos válidos.
PostCondiciones de Exito	En la tabla articulos se observan nuevos artículos y articulos eliminados
PostCondiciones de Fallo	Código de nuevo articulo no exista. Código de articulo eliminado esté activo. Código no válido para consulta.
Actores Primarios	Vendedor – Almacén
Actores Secundarios	Tabla de Artículos (BD)
Trigger	Petición dc requerimiento: crear, eliminar o consultar artículos.

Descripción	Pasos	Acción
	1.1	Se hace requerimiento de crear nuevo articulo.
	1.2	Sistema genera siguiente código disponible, se ingresan descripción, marca y costo.
	1.3	Se actualiza articulo y se incrcmenta el **stock**
	2.1	Se hace requerimiento de eliminar articulo.
	2.2	Se ingresa cbdigo para consultar.
	2.3	Si no hay stock se elimina el artículo
	3.1	Se hace requerimiento de consultar artículo
	3.2	Se ingresa el código de artículo
	3.3	Muestra los resultados en pantalla

Extensiones	Pasos	Acción
	1	Al crear nuevo artículo mostrar en pantalla si esta repetido; al eliminar articulo mostrar mensaje en pantalla si no se tuvo éxito; al consultar articulo mostrar en pantalla si no es válido.

Precio de Venta del Artículo

Caso de Uso	Alcance
Meta	Registrar precio **de** venta del artículo
Alcance	Sistema
Nivel	Primario
PreCondiciones	Código y costo del articulo debe existir
PostCondiciones de Exito	El artículo ha actualizado su precio.
PostCondiciones de Fallo	El artículo no tiene costo.
Actores Primarios	Vendedor – Almacén
Actores Secundarios	Tabla de artículos (BD)
Trigger	Petición de registrar precio de venta del articulo.

Descripción	Pasos	Acción
	1	Se hace requerimiento de registrar precio de venta
	2	Ingresar el precio de venta **del** artículo.
	3	Registrar el precio de venta

Extensiones	Pasos	Acción
	1	Si no se ha ingresado el costo del artículo, mostrar mensaje en pantalla.

Manejo de Clientes

Caso de Uso	Alcance
Meta	Crear, eliminar y consultar clientes
Alcance	Sistema
Nivel	Primario
PreCondiciones	1.- Para crear un nuevo cliente, este no debe de existir. 2.- Se eliminan sólo los clientes si no tienen facturas pendientes. 3.- Se consulta sólo los clientes activos y con códigos válidos.
PostCondiciones de Exito	En la tabla clientes ***se*** observan nuevos clientes y clientes eliminados
PostCondiciones de Fallo	Código de nuevo cliente no exista Código de cliente eliminado esté activo. Código no válido para consulta.
Rctores Primarios	Vendedor - Almacén
Actores Secundarios	Tabla de Clientes (BD)
Trigger	Petición de requerimiento: crear, eliminar ***o*** consultar clientes.

Descripción	Pasos	Acción
	1.1	Se hace requerimiento de crear nuevo cliente.
	1.2	Sistema genera siguiente código disponible, se ingresan nombres, apellidos, dirección y teléfono.
	1.3	Se actualiza clicnte.
	2.1	Se hace requerimiento de eliminar cliente.
	2.2	Se ingresa código para consultar.
	2.3	Si no hay factura pendiente se elimina el cliente
	3.1	Se hace requerimiento ***de*** consultar cliente
	3.2	Se ingresa el código de cliente
	3.3	Muestra los resultados en pantalla

Extensiones	Pasos	Acción
	1	Al crear nuevo cliente mostrar en pantalla si está repetido; al éxito; al consultar cliente mostrar en pantalla si no es válido.

2.4.2 Identificación de Objetos:

Atributos

Factura

- No_factura — Numeración de factura
- Código_cliente — Código del cliente
- Fecha_factura — Fecha de emisión de factura
- Total_factura — Valor a pagar de la factura incluido el iva
- Código_artículo — Código del articulo
- Cantidad_vendida_artículo — Cantidad vendida del artículo
- Cantidad_devuelta_artículo — Cantidad devuelta del articulo vendido
- Estatus_factura — Estado de la **factura** (V: vigente ó M modificada)

Artículo

- Código_artículo — Código del articulo
- Precio_artículo — Precio de venta del articulo
- Costo_artículo — Costo del articulo

- Descripción_artículo — Descripción del articulo
- Marca_artículo — Marca del artículo
- Stock_artículo — Cantidad existente en stock del artículo
- Tipo_ajuste — Motivo ajuste del artículo (R:Robo,B:Baja,D:Daño)
- Estatus_articulo — Estado del articulo **(A:** activo 6 E: eliminado)

Cliente

- Código_cliente — Código del cliente
- Nombres_cliente — Nombres del cliente
- Apellidos_cliente — Apellidos del cliente
- Dirección_cliente — Dirección del cliente
- Estatus_cliente — Estado del cliente **(A:** activo 6 E: eliminado)

Vendedor

- Código_vendedor — Código del vendedor
- Nombres_vendedor — Nombres del vendedor
- Apellidos_vendedor — Apellidos del vendedor
- Dirección_vendedor — Dirección del vendedor
- Estatus_vendedor — Estado del vendedor **(A:** activo ó E: eliminado)

CIB a ESPOL

Métodos

NuevoCódigoFactura.- Asigna el siguiente código de factura disponible para la venta del artículo.

GuargarDatosFactura.- Registra la cabecera y detalle de la factura en la base.

ValorPagar.- Calcula el subtotal de la venta incluido el iva **para** obtener el total de la factura a cancelar.

ConsultaFactura.- Captura el código de la factura, realiza la búsqueda en la base de datos y la muestra los resultados en pantalla.

PagoFactura.- Registra el pago de la factura en la tabla ventas.

ActualizaFactura.- Actualiza la factura cuando se ha realizado alguna devolución de artículo ya vendido.

ActualizaVentas.- Actualiza la tabla ventas cuando se ha realizado alguna devolución de artículo ya vendido.

ConsultaVendedor.- Muestra los datos del vendedor en pantalla según el código ingresado.

NuevoCódigoCliente.- Asigna el siguiente código de cliente disponible para la base de datos.

NuevoCliente.- Registra el nuevo cliente en la tabla cliente de la base de datos.

EliminaCliente.- Elimina los datos de un cliente en la tabla cliente de la base de datos.

ConsultaCliente.- Muestra los datos del cliente en pantalla según el código ingresado.

NuevoCódigoArtículo.- Asigna el siguiente código de articulo disponible para la base de datos.

NuevoArtículo.- Registra el nuevo articulo en la tabla artículo de la base de datos.

EliminaArtículo.- Elimina los datos de un artículo en la tabla articulo de la base de datos.

ConsultaArtículo.- Muestra los datos del articulo en pantalla según el código ingresado.

ConsultaStock.- Muestra el stock del artículo en pantalla según el código ingresado.

IncrementaStock.- Incrementa el stock del artículo en la base.

ActualizaStock.- Actualiza el stock del articulo ya sea por compra, ajuste ó devolucibn.

2.5 Diseño

Como resultado del análisis a través de los casos de usos, se identificaron **4** objctos en el sistema, los mismos que son:

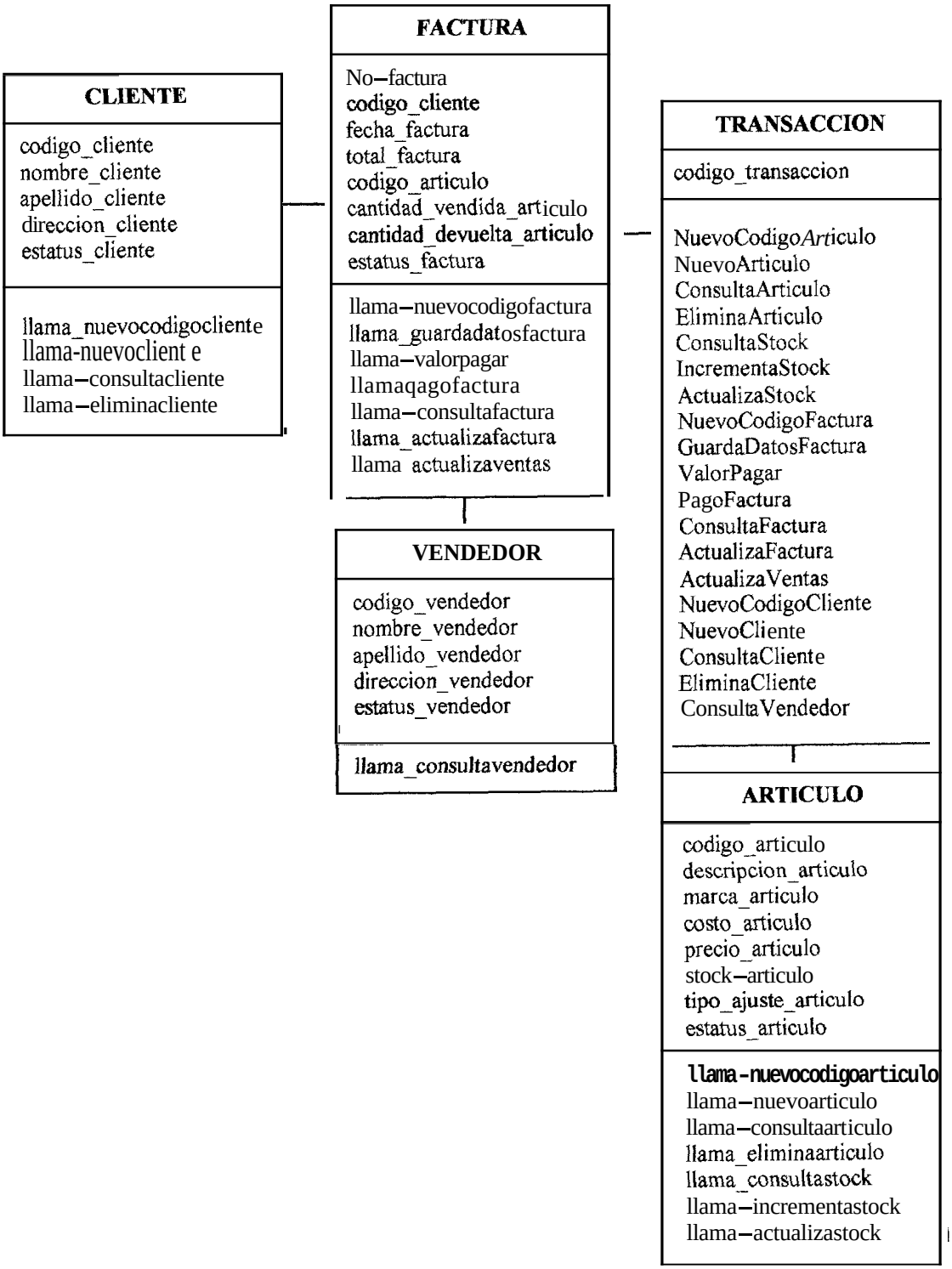
1. Artículo
2. Factura
3. Cliente
4. Vendedor

Debido a que el sistema está orientado a manejar transacciones, cada uno de los 4 objetos deberk accesar a la base de datos y como la base ha sido implementada en el sistema SQL Server aumenta un objeto más, para ejecutar cada una de las diferentes transacciones mediante sentencias SQL. A este nuevo objeto lo hemos llamado: Objeto Transacción.

Se tiene entonces 5 objetos para implementar el sistema Venta de Electrodomésticos, estos son:

1. Artículo
2. Factura
3. Cliente
4. Vendedor
5. Transacción

2.5.1 Modelo de Objetos



2.6 Implementación

El Sistema ha sido desarrollado utilizando la invocación estática, debido a que los objetos identificados fueron sólo **4,** siendo utilizados frecuentemente, asi para que el sistema tenga un tiempo de respuesta aceptable, es ideal que la interface IDL se cncuentre previamente compilada y que los objetos: artículo, factura, cliente y vendedor se encuentren previamente cargados en el repositorio de implcmentacicin.

Las herramientas dc programación quc se utilizaron heron:

JDK 1.1.7, Visual Café 2.1, Visibroker **3.4,** Ncstcape Communicator **4.6** y SQL Server 7.0

Capítulo 3

IIOP

3.1 Protocolos para Aplicaciones C/S en Internet

La comunicación de datos se ha convertido en una parte fundamental de la computacibn. Las redcs globales reúnen datos sobre temas diversos, como las condiciones atmosféricas, la produccibn de cosechas y el tráfico adreo. Algunos grupos establecen listas de correo electrónico para poder compartir informacicin de interés común. Las personas que tienen pasatiempo intercambian programas para sus computadoras personales. En el mundo cientifico, las redes de datos son esenciales pues permiten a los cientificos enviar programas y datos hacia supercomputadoras remotas para su procesamiento, rccuperar los resultados e intercambiar informacicin con sus colegas.

Una red que está conectada a Internet depende totalmente de una serie de protocolos llamados en conjunto TCP/IP, los cuales mueven informacicin a través de la red, cada uno de dstos protocolos transfiere datos usando formatos diferentes y con distintas opciones.

El protocolo TCP/IP ha sido utilizado por mucho tiempo para la comunicación entre redes dentro de una corporación, aún cuando las empresas no

tengan una conexión hacia redes externas. Otros grupos utilizan éste protocolo para comunicarse entres sitios geográficamente alejados unos de otros.

Aunque la tecnologia TCP/IP es significativa por sí misma, es espccialmcntc interesante debido a que su viabilidad ha sido demostrada a gran escala. Esta forma la tecnologia base para una red de redes global que conecta hogares, campus universitarios y otras escuelas, corporaciones y laboratorios gubernamentales en diferentes países.

3.2 TCP/IP

La mayor parte de las aplicaciones necesitan mucho mas que sólo la entrega de paquetes, debido a que se requieren que el software de comunicaciones se recupera de manera automática de los errores de transmisión, paquetes perdido o fallas de conmutadores intermedios a lo largo del camino entre el transmisor y el receptor. El servicio de transporte confiable TCP/IP (Transmission Control Protocol / Internet Protocol) resuelve estos problemas. Permite que una aplicación en una computadora establezca una "conexión" con una aplicación en otra computadora, para después enviar un gran volumen de datos a través de la conexicin como si ésta fuera permanente y directa del hardware. Debajo de todo esto, por supuesto los protocolos de comunicación dividen el flujo de datos en pequeños mensajes y los

envían, uno tras otro, esperando que el receptor proporcione un acuse de recibo de la recepción.

3.2.1 IP

La entrega sin conexión es una abstracción del servicio que la mayoría de las redes de conmutación de paquetes ofrece. Simplemente significa que una red de redes TCP/IP rutea mensajes pequeños de una máquina a otra, basándose en la información de dirección IP (Internet Protocol) que contiene cada mensaje. Debido a que el servicio sin conexión rutea cada paquete por separado, no garantiza una entrega confiable y en orden. Como por lo general se introduce directamente en el hardware subyacente, el servicio conexión como la base de todos los servicios de red de redes, hace que los protocolos TCP/IP sean adaptables a un amplio rango de hardware de red.

En otras palabras una dirección de Internet es una dirección IP. La mayoría de los usuarios e incluso la literatura sobre Internet asocian las direcciones IP a las computadoras anfitrión, pero la computadora en realidad no tiene una dirección IP. Una red TCP/IP asocia las direcciones IP con una tarjeta de interface, no con la computadora anfitrión. Si consideramos que una computadora puede tener varias tarjetas de interface de red, significa que una computadora anfitrión puede tener varias direcciones IP válidas.

Una dirección IP válida es de 32 bits o 4 bytes de longitud. Aunque existen varios tipos de notación, pero la más utilizada es la "notación decimal", la cual representa una serie de números decimales separados por puntos (Ej: 100.100.100.100).

3.2.2 Sockets

Es una interface de programas de aplicación (API) para redes TCP/IP. Una API es un grupo de funciones que emplean los programadores a fin de desarrollar aplicaciones para un ambiente de cómputo específico. A ésta API también se la conoce como **Interface de Sockets de Berkeley.**

Un socket es una interface para la comunicación por red. Es una representación abstracta en el proceso de comunicación. Involucra a dos computadoras anfitriones o procesos que se pasan datos entre sí a través de la red. Cuando un programa utiliza la interface de sockets para una comunicación en red, necesita un sockets en cada extremo de la comunicación para poder " conversar ", el un extremo es el local y el otro es el remoto. Analizando este proceso desde el punto de vista del modelo cliente/servidor, se necesita un sockets para la aplicación que inicia la búsqueda activa (cliente) y otro para la aplicación pasiva (servidor). Cuando se necesita un sockets se define sus características y se utiliza una API para solicitar al software de red un identificador que reconozca al sockets especificado. Los

programas basados en sockets crean un sockets y entonces como paso aparte, lo conectan a un extremo destino. Los sockets pueden ser orientados a conexión o sin conexión. Un proceso de comunicación usando sockets utiliza el modelo básico de abrir - leer - escribir – cerrar (figura 3.1).

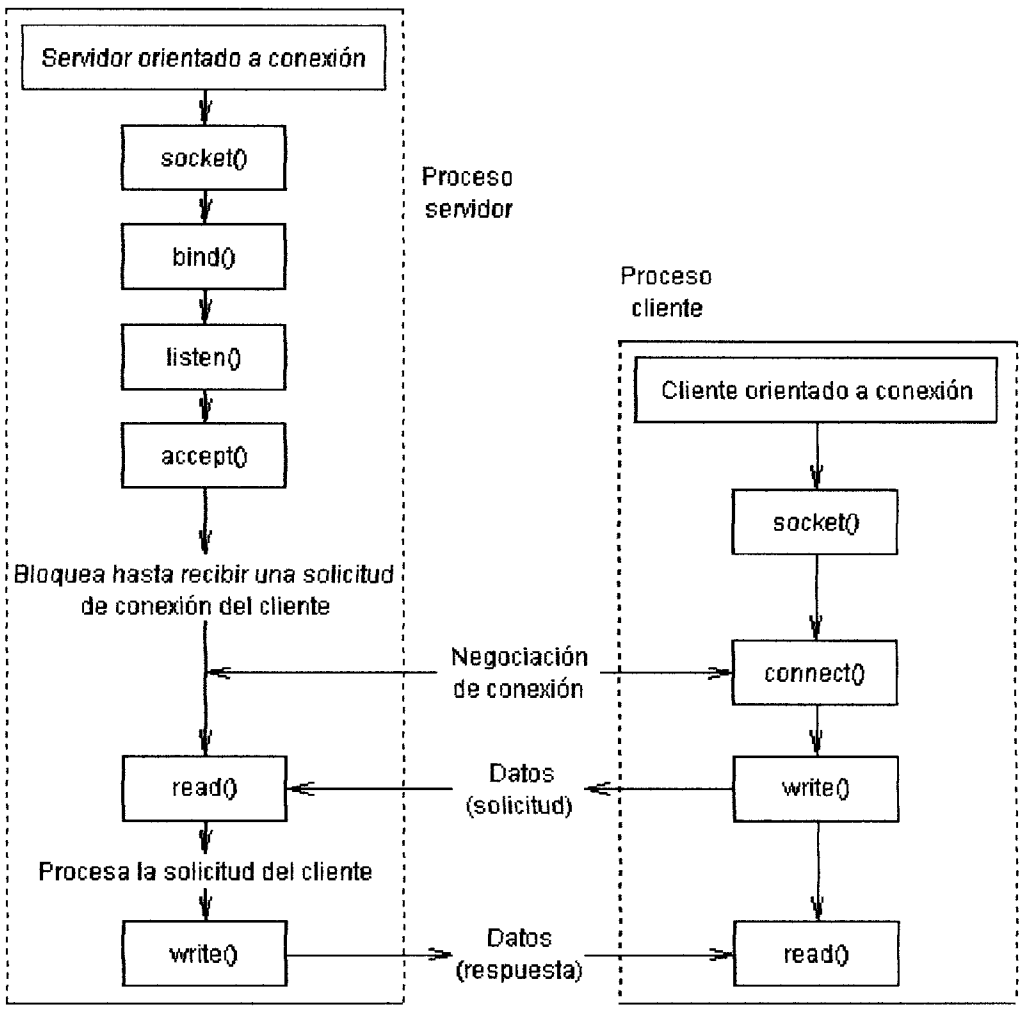


figura 3.1

3.3 IIOP

El Internet Inter-ORB Protocol (IIOP) es un protocolo estándar que permite la interoperabilidad en la comunicación entre 2 ORB. Es decir, que un cliente ORB de un vendedor A puede invocar métodos de un objeto implementado a otro servidor ORB de un vendedor B.

El protocolo IIOP reemplaza al protocolo TCP/IP para lograr la comunicación entre aplicaciones Cliente/Servidor en Internet, ya que este protocolo se lo puede ubicar como otro servicio en la capa de transporte.

IIOP establece la comunicación entre 2 ORB sin importar la dirección IP de la computadora, es decir que los requerimientos no son enviados en forma de paquetes, tan sólo se basa en enviar objetos distribuidos al ORB cliente para que pueda invocar sus métodos como si los tuviera en su misma máquina convirtiéndose en una invocación local.

3.4 ¿Cómo trabaja IIOP en Internet?

Basados en los principios de hypertext, la tecnología WWW (World Wide Web) fue diseñada para distribución y presentación de datos. Ciertamente más tarde aparecieron las páginas dinámicas, para el envío de datos desde el cliente hasta el servidor, con este propósito fue introducido el GCI (Gateway Commom Interface). Luego aparece PERL, del lado del servidor con una interface tediosa y difícil de

implementar. Adicionalmente, los diálogos dinámicos entre cliente/servidor eran imposibles debido a un bloqueo natural para transferir datos. En este punto entra en escena el IIOP.

Con IIOP se tienen las siguientes ventajas:

- El cliente y servidor se comunican con mucho menos overhead, es decir que se evita el cuello de botella HTTP/CGI.
- Los objetos del lado del servidor pueden ser implementados en cualquier lenguaje.
- Pueden ser intercambiados tipos de datos reales y no solamente strings.
- Con CORBA e IIOP, los detalles de administración de la interface y mecanismos de paso de parámetros son automáticos.
- El código que maneja la codificación, paso y decodificación de parámetros es generado automáticamente, no así con HTML/HTTP.
- Las aplicaciones bajo CORBA e IIOP son de más fácil mantenimiento, a la vez que tienen componentes reusables de aplicación.

3.4.1 Conexión IIOP entre un Cliente browser y un Servidor CORBA

1. Un browser descarga páginas HTML e incluye referencias a applets Java.

2. El browser obtiene el applet Java desde el servidor HTTP y es descargado al browser en forma de bytecodes.
3. El browser carga el applet, en memoria.
4. El applet invoca objetos CORBA vía IIOP.
5. Los objetos servidores puede generar la siguiente página HTML para éste cliente, pero dinámicamente.

Capítulo 4

Base de Datos

4.1 Administrador de Base de Datos (DBMS)

Database Management System "Sistema Administrador de Base de Datos" (DBMS), es un proceso servidor cuyo objetivo principal es proveer datos ó servicios a procesos clientes que los soliciten.

El Servidor de Base de Datos, a veces también llamado "Motor SQL", proporciona las vistas lógicas y físicas de los datos, administra el control y ejecución de comandos SQL, permite que múltiples procesos clientes accedan a la base al mismo tiempo y provee un entorno que protege a la base.

4.1.1 SQL

SQL (Structured Query Languaje "Lenguaje Estructurado de Consulta"), es una herramienta que permite organizar, administrar y recuperar datos almacenados en una base de datos informático.

SQL es el lenguaje que se utiliza para interactuar con una base de datos, particularmente con base de datos relacionales.

Cuando es necesario recuperar datos de la BD, la petición se realiza utilizando SQL. EL DBMS procesa la petición SQL, recoge los datos solicitados y los devuelve a quien los solicitó. Este procedimiento de petición de datos y posterior recepción de resultados se llama *"query"*.

SQL se utiliza para controlar todas las funciones de un DBMS lo cual incluye:

- **Definición de datos.-** Permite que un usuario defina la estructura y la organización de los datos almacenados así como las relaciones existentes entre ellos.
- **Recuperación de datos.-** Permite que un usuario ó a un programa recuperar y utilizar los datos almacenados en una base de datos.
- **Manipulación de datos.-** Permite a un usuario ó a un programa actualizar la base de datos añadiendo datos nuevos, borrando los viejos y modificando los almacenados previamente.
- **Control de acceso.-** Permite restringir la capacidad de un usuario para recuperar, añadir y modificar datos, protegiendo los datos almacenados contra accesos no autorizados.
- **Compartición de información.-** Permite coordinar la compartición de datos entre usuarios concurrentes asegurando que no haya interferencia entre ellos.

- **Integridad de datos.**- Permite definir restricciones de integridad en la base de datos protegiéndola de alteraciones debidas a actualizaciones inconsistentes o fallas del sistema.

4.1.1.1 Funciones que desempeña SQL

- **Es un lenguaje interactivo de consultas**.- Permite a los usuarios de manera interactiva recuperar datos y presentarlos en pantalla.
- **Es un lenguaje de programación de base de datos**.- Permite a los programadores introducir órdenes SQL de sus programas para acceder a los datos.
- **Es un lenguaje de administración de base de datos**.- Permite al administrador de una minicomputadora o sistema basado en una computadora central utiliza SQL para definir la estructura de la base de datos y el control de acceso a los datos almacenados.
- **Es un lenguaje para arquitectura C/S**.- Los programas de las computadoras personales utilizan SQL para comunicarse, a través de una red de área local, con los servidores de bases de datos que almacenan los datos compartidos. Muchas nuevas aplicaciones utilizan ésta arquitectura C/S, que minimiza el tráfico de la red.
- **Es un lenguaje de base de datos distribuidas**.- Los sistemas de administración de base de datos distribuidas utilizan SQL para ayudar a

distribuir los datos a través de muchos sistemas informáticos conectados. El DBMS de cada sistema utiliza SQL para comunicarse con el resto de sistemas, enviando peticiones para el acceso a los datos.

4.2 Java Database Connectivity (JDBC)

El JDBC API define clases Java para realizar conexiones a bases de datos, sentencias SQL, conjunto de resultados, etc. Esto permite al programador ejecutar sentencias SQL y procesar los resultados. JDBC es la primera API para acceso basándose en datos en Java.

El API JDBC es implementado para manejarlo como drives que soporta múltiples drives de conexiones a diferentes base de datos. Los drives JDBC pueden ser enteramente escritos en Java que pueden ser bajados como parte del applet, o pueden ser implementados usando métodos para puentear con librerías de acceso de las bases de datos existentes.

4.3 Acceso a Base de Datos JDBC/ODBC

Open Database Connectivity (ODBC), controlador de Microsoft Windows que tiene un API estándar para realizar conexiones a Base de Datos, sentencias SQL, conjunto de resultados, llamadas a funciones (CLI), etc.

Cuando se tiene una aplicación Java y se quiere accesar a una BD SQL Server, se debe puentear para poder conectarse con la base. Este puente se lo logra mediante el enlace JDBC/ODBC (figura 4.1).

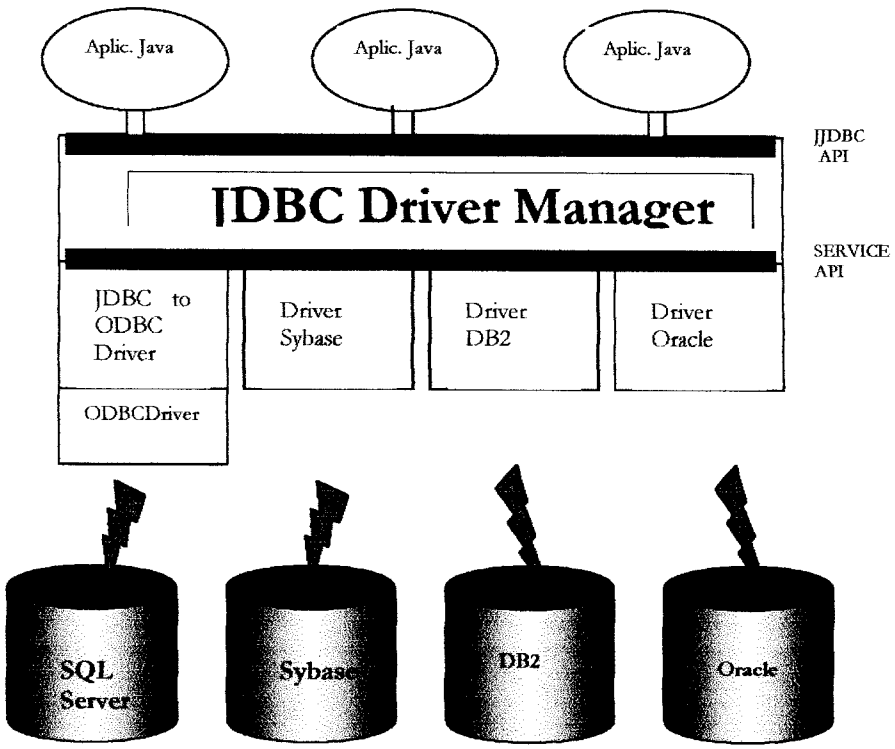


figura 4.1

Así en el código Java se debe de incluir las siguientes líneas:

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();

DriverManager.getConnection(url,usuario,clave);

Donde :

Url corresponde al nombre de la base de datos definido en el ODBC.

Usuario corresponde al user definido en el ODBC.

Clave corresponde al password definido en el ODBC.

Capítulo 5

Aplicación para Empresas Comerciales

5.1 Uso

El Sistema Venta de Electrodomésticos va dirigido a pequeñas y medianas empresas del sector comercial que se dedican a la venta de productos electrodomésticos y venderlos a usuarios finales.

El Sistema está ideado para ser utilizado a través de Internet, lo cual permite que el cliente pueda realizar sus compras desde su hogar u otro sitio por intermedio de vendedores, sin necesidad de que acerque al almacén.

5.2 Arquitectura Abierta

El diseño del sistema ha sido desarrollado pensando en la amplitud del mismo, es decir que es totalmente abierto y flexible para realizar cambios según los requerimientos y necesidades de la empresa.

La flexibilidad del Sistema es una de las principales ventajas que se tiene cuando este ha sido diseñado orientado a objetos.

Cada uno de los 4 objetos (artículo, factura, cliente y vendedor) nos permiten manejar de manera independiente atributos y métodos para ampliar las funciones de cada uno.

En cuanto a las transacciones podría también ampliarse sin ningún problema, ya que son manejados a través de un objeto totalmente independiente de los 4 objetos anteriormente mencionado, éste objeto se llama transacción.

Estas facilidades nos permite sacarle el mayor provecho a un sistema que utiliza la arquitectura CORBA, ya que los objetos además de ser independientes se encuentran distribuidos, es decir que pueden estar localizados en cualquier punto de la red. Así un objeto cliente puede convertirse en un objeto servidor y viceversa dependiendo de la ocasión.

En resumen todas estas ventajas se obtienen con respecto al Sistema, ya que CORBA es el mejor middleware cliente/servidor definido hasta ahora.

Debemos recordar que los objetos de CORBA pueden inter-operar con objetos sobre otras plataformas y además pueden estar escritos en cualquier lenguaje de programación.

Conclusiones y Recomendaciones

Sobre la base de lo estudiado y a los objetivos planteados inicialmente, se pueden obtener las siguiente conclusiones y recomendaciones.

Conclusiones

- Implementar el proyecto utilizando la técnica Programación Orientado a Objetos fue difícil, ya que en el transcurso del estudio de la carrera para elaborar los proyectos la técnica usada era la Programación Estructurada.
- Luego de utilizar la metodología orientada a objetos, descubrí que implementar el Sistema sería mucho más fácil que usar la metodología estructurada. Así, para poder realizarles cambios y darle mantenimiento al Sistema es más fácil.
- Utilizar la tecnología CORBA se tiene totalmente independencia entre los objetos.
- Los objetos de CORBA son entidades inteligentes que pueden vivir y descubrirse en cualquier punto de una red.
- La ambiciosa meta de CORBA es "IDLizar" todo el middleware cliente/servidor y todos los componentes residentes en un ORB. OMG espera alcanzar esta meta siguiendo dos pasos: 1) Convertirlos todo en clavo y 2) Darles a todos un martillo.

El "clavo" es el IDL de CORBA. Este permite a los proveedores de componentes especificar en un lenguaje de definición estándar la interfaz y estructura de los objetos que ofrecen.

El "martillo" incluye el conjunto de servicios distribuidos que brindarán los proveedores de OMG. Estos servicios determinarán qué objetos habrán de encontrarse en la red, cual métodos proporcionarán y qué adaptadores de interfaces de objetos soportarán.

- El ORB de CORBA, permite invocar métodos de objetos del servidor con el lenguaje de alto nivel que se prefiera puede ser: C, C++, Ada, Java, Smalltalk, etc. No importa en que lenguaje hayan sido generados los objetos del servidor, CORBA distinguen entre la interface y el objeto implementado.
- EL ORB, permite definir estáticamente invocaciones de métodos en tiempo de compilación, ó bien descubrirlas dinámicamente en tiempo de ejecución. Depende únicamente de la cantidad de objetos, es decir que si son pocos es mejor usar invocación estática y si son millares de objetos donde tan sólo se necesiten más a unos que a otros habría que usar la invocación dinámica.
- Existe una transparencia local/remota, es decir que un ORB puede correr en modo autónomo en una portátil ó interconectarse con todos los demás ORB. Puede servir de intermediario para llamadas entre objetos, dentro de un solo proceso, en procesos múltiples en ejecución dentro de la misma máquina ó en procesos múltiples en ejecución entre redes y sistemas operativos.

- CORBA permite seguridad y transacciones integradas, es decir que el ORB incluye en sus mensajes información de contexto para el manejo de seguridad y transacciones entre fronteras de máquinas y ORB.
- CORBA usado en internet disminuye el cuello de botella entre el cliente y el servidor, es decir que el requerimiento puede ser construido por una serie de objetos y no enviar uno a uno al servidor.

Recomendaciones

a) De búsqueda de información

Si desea tener mayor conocimiento sobre alguno de los temas expuestos en este documento diríjase a la sección Bibliografía.

b) De mejoramiento de la aplicación

Como el Sistema ha sido diseñado para ser utilizado en Internet, podría ser utilizado no solo por vendedores virtuales sino por los clientes, es decir desde su hogar u otro sitio, en este caso se le restringiría ciertas opciones de administración del Sistema y únicamente se le habilitarían opciones de compra del producto.

c) De equipos de computación y comunicaciones.

Para utilizar el sistema Venta de Electrodomésticos, se necesitan por lo menos 2 computadoras, una para el servidor y otra para el cliente; y una infraestructura de comunicaciones como se detalla a continuación:

Servidor :

Para tener un buen tiempo de respuesta de la aplicación servidor hacia los requerimientos de los clientes, se recomienda que el equipo tenga las siguientes características mínimas:

- Equipo multiprocesador.
- Procesadores de 64 bits con velocidades de 300 MHz.
- Memoria RAM de 128 Mb.
- Disco duro tipo SCSI de 3 Gb.
- Monitor SVGA .28 de 17"
- Tarjeta de vídeo de 4Mb

Cliente :

Se requiere de un potencial de procesamiento medio, puede tener las siguiente características:

- Equipo monoprocesador.
- Procesadores de 32 bits con velocidades de 233 MHz.
- Memoria RAM de 64 Mb.
- Disco duro de 1 Gb.
- Monitor SVGA .28 de 15"
- Tarjeta de vídeo de 4Mb

Comunicaciones :

- Tarjeta de red de 100Mbps
- Medio físico de transmisión UTP extensión 5 ó fibra óptica (recomendado)
- Concentradores / Ruteadores de 100 Mbps
- Tarjeta de módem de 56.6K (PCI)
- Línea de teléfono
- Cuenta de Internet de algún proveedor

ANEXOS

Anexo 1

Base de Datos Artículos – Tablas

Tabla artículo

Column Name	Datatype	Lenght	AllowNulls	Identity
cod_articulo	int	4		√
des_articulo	varchar	50		
mar_articulo	varchar	10		
pre_unitario	int	4		
iva_articulo	int	4		
est_articulo	char	1		
can_disponible	int	4		
cos_articulo	int	4		

Tabla cab_factura

Column Name	Datatype	Lenght	AllowNulls	Identity
cod_factura	int	4		√
cod_vendedor	int	4		
cod_cliente	int	4		
fec_emision	floa	8		
est_cabecera	char	1		
tot_factura	int	4		

Tabla det_factura

Column Name	Datatype	Lenght	AllowNulls	Identity
cod_factura	int	4		
cod_articulo	int	4		

pre_unitario	int	4	
can_vendida	int	4	
can_devuelta	int	4	√
pre_subtotal	int	4	
est_detalle	char	1	
cos_articulo	int	4	

Tabla cliente

Column Name	**Datatype**	**Lenght**	**AllowNulls**	**Identity**
cod_cliente	int	4		√
nom_cliente	varchar	30		
ape_cliente	varchar	30		
dir_cliente	varchar	70		
tel_cliente	varchar	10	√	
est_cliente	char	1		

Tabla vendedor

Column Name	**Datatype**	**Lenght**	**AllowNulls**	**Identity**
cod_vendedor	int	4		√
nom_vendedor	varchar	25		
ape_vendedor	varchar	25		
dir_vendedor	varchar	70		
tel_vendedor	varchar	10	√	
sue_vendedor	int	4	√	
com_vendedor	int	4	√	
fec_desde	float	8	√	
fec_hasta	float	8	√	

sta_vendedor	char	1		
mon_vendedor	int	4	√	

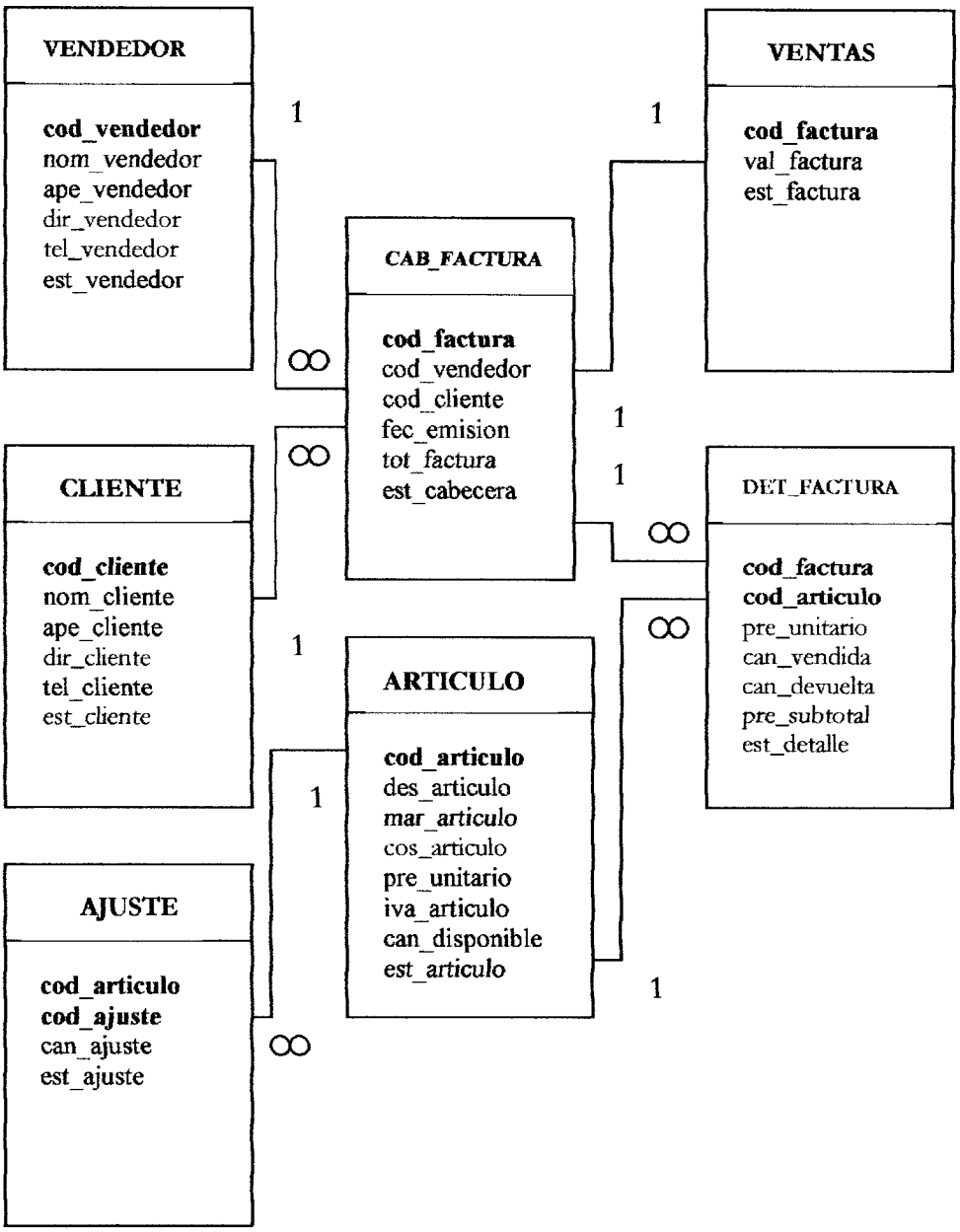
Tabla ventas

Column Name	**Datatype**	**Lenght**	**AllowNulls**	**Identity**
cod_factura	int	4		
val_factura	int	4		
est_factura	char	1		

Tabla ajuste

Column Name	**Datatype**	**Lenght**	**AllowNulls**	**Identity**
cod_ajuste	numeric	9		√
cod_articulo	int	4		
can_ajustada	int	4		
est_ajuste	char	1		

Modelo Entidad – Relación



Anexo 2

Código de los Módulos Principales

Código Interface IDL

```
module Ventas{
  interface Articulo{
      attribute long cod_articulo;
      attribute wstring des_articulo;
      attribute wstring mar_articulo;
      attribute long long pre_articulo;
      attribute long iva_articulo;
      attribute wchar est_articulo;
      attribute long can_act_articulo;
      attribute long can_inc_articulo;
      attribute long can_ven_articulo;
      attribute wchar mot_aju_articulo;
      attribute long can_aju_articulo;
      boolean CierraConexion();
      wstring NuevoCodigoArticulo(in short TipoCodigo);
      wstring NuevoArticulo(in long CodArticulo,in string
                        DesArticulo,in string MarArticulo,
                        in long long PreArticulo,in long IvaArticulo);
      wstring ConsultarArticulo(in long CodArticulo);
      wstring ConsultarTodosArticulos();
      wstring EliminarArticulo(in long CodArticulo);
      wstring NuevoPrecioArticulo(in long CodArticulo,in long long
                                  NuePrecio);
     wstring IncrementarStock(in long CodArticulo,in long
                                CantIncrementar);
     wstring DevolucionArticulo(in long CodArticulo,in long
                                CodFactura,in long CanDevolver);
     wstring AjusteArticulo(in long CodArticulo,
                            in wchar MotAjuste,in long CanAjuste);
      wstring ConsultarStock(in long CodArticulo);
  };

  interface Factura {
     attribute long cod_factura;
      attribute wchar estatus_cabecera_factura;
      attribute wchar estatus_detalle_factura;
      attribute double fecha_creacion;
      boolean CierraConexion();
      wstring NuevoCodigoFactura(in short TipoCodigo);
      wstring NuevaCabeceraFactura(in long CodFactura,in long
                  CodCliente,in long CodVendedor,in long long PreTotal,
                   in double FecEmision);
      wstring NuevaDetalleFactura(in wstring NueDetalles);
      wstring ConsultaCabeceraFactura(in long CodFactura);
      wstring ConsultaDetalleFactura(in long CodFactura);
      wstring PagoFactura(in long CodFactura,in long long ValFactura);
      wstring ActualizarPago(in long CodFactura,in long long
                                NueValFactura);
  };
```

```
    interface Cliente {
       attribute long cod_cliente;
       attribute wstring nom_cliente;
       attribute wstring ape_cliente;
       attribute wstring dir_cliente;
       attribute wstring tel_cliente;
       attribute wchar estatus_cliente;
       boolean CierraConexion();
       wstring NuevoCodigoCliente(in short TipoCodigo);
       wstring NuevoCliente(in long CodCliente,in wstring NomCliente,in
                       wstring  ApeCliente,in wstring DirCliente,
                       in wstring TelCliente);
       wstring ConsultarCliente(in long CodCliente);
       wstring ConsultarTodosClientes();
       wstring EliminarCliente(in long CodCliente);
    };

    interface Vendedor  {
       attribute long cod_vendedor;
       attribute wstring nom_vendedor;
       attribute wstring ape_vendedor;
       attribute wstring dir_vendedor;
       attribute wstring tel_vendedor;
       attribute long long sue_vendedor;
       attribute wstring use_vendedor;
       attribute wstring pas_vendedor;
       attribute wchar estatus_vendedor;
       attribute double fec_desde;
       attribute double fec_hasta;
       boolean CierraConexion();
       wstring NuevoCodigoVendedor(in short TipoCodigo);
       wstring NuevoVendedor(in long CodVendedor,in wstring
              NomVendedor,in wstring ApeVendedor,in wstring DirVendedor,
              in wstring TelVendedor,in long long SueVendedor);
       wstring ConsultarVendedor(in long CodVendedor);
       wstring ConsultarTodosVendedores();
       wstring EliminarVendedor(in long CodVendedor);
       wstring MontoVenta(in long CodVendedor,in double FechaDesde,in
             double FechaHasta);
       wstring PagoComision(in long CodVendedor,in double MontoVenta,in
             short PorGanancia,in double Total,in double FechaDesde,
    in double FechaHasta);
     };
};
```

Código de Objetos Implementados

El código de los objetos implementados sólo se presenta a ArticuloImpl, únicamente para mostrar como es la estructura de los mismos.

Código Objeto ArticuloImpl

```
import ArticuloBD.*;
import java.lang.*;
```

```
public class ArticuloImpl extends Ventas._ArticuloImplBase {
  ArticuloBD  articulobase = new ArticuloBD();
  String url = "jdbc:odbc:articulos";
  String usuario = "alex";
  String clave = "alex";

  public ArticuloImpl(java.lang.String name) {
      super(name);
      System.out.println("Objeto Articulo Creado");
}
public boolean CierraConexion()
{ //--------------   Seccion de cierre de conexion   ------------------
     return(articulobase.Cerrar_base());
}

public java.lang.String NuevoCodigoArticulo(short TipoCodigo)
{//--------------   Seccion de apertura de conexion   ----------------
     try{    //realiza una conexion a base
        if(articulobase.Conectar(url,usuario,clave)){
           System.out.println("Conectado para manejar articulo");}
        else{
           System.out.println("Error en la conexion a la base de datos");
           return "-15";}
      }catch (Exception ee){
        System.out.println("Error al conectarse a la base desde
                            articulo");}
//--------------   Seccion de ejecucion query   ----------------------
  String res_cadena = articulobase.NuevoCodigo(TipoCodigo);
//--------------   Seccion de cierre de conexion   -------------------
  boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   ----------------
  return(res_cadena);
}

public java.lang.String NuevoArticulo(int CodArticulo,java.lang.String
         DesArticulo,java.lang.String MarArticulo,long PreArticulo,
         int IvaArticulo)
{//--------------   Seccion de apertura de conexion   ----------------
    try{    //realiza una conexion a base
     if (articulobase.Conectar(url , usuario , clave)){
        System.out.println("Conectado para manejar articulo");}
     else{
        System.out.println("Error en la conexion a la base de datos");
        return "-15";}
    }catch (Exception ee){
       System.out.println("Error al conectarse a la base desde
                           articulo");}
//-------------   Seccion de ejecucion query   -------------------
  String res_cadena=articulobase.NuevoArticulo(CodArticulo,DesArticulo,
  MarArticulo,PreArticulo,IvaArticulo);
//--------------   Seccion de cierre de conexion   ----------------
  boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   -------------
  return(res_cadena);
}
public java.lang.String ConsultarArticulo(int CodArticulo)
{//--------------   Seccion de apertura de conexion   ---------------
    try{    //realiza una conexion a base
       if (articulobase.Conectar(url , usuario , clave)){
          System.out.println("Conectado para manejar articulo");}
       else{
```

```
          System.out.println("Error en la conexion a la base de datos");
          return "-15";}
     }catch (Exception ee){
       System.out.println("Error al conectarse a la base desde
                           articulo");}
//--------------   Seccion de ejecucion query   -------------------
   String res_cadena = articulobase.ConsultarArticulo(CodArticulo);
//--------------   Seccion de cierre de conexion   ----------------
   boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   --------------
   return(res_cadena);
}

public java.lang.String ConsultarTodosArticulos()
{//--------------   Seccion de apertura de conexion   ---------------
   try{    //realiza una conexion a base
     if (articulobase.Conectar(url , usuario , clave)){
       System.out.println("Conectado para manejar articulo");}
     else{
       System.out.println("Error en la conexion a la base de datos");
       return "-15";}
   }catch (Exception ee){
       System.out.println("Error al conectarse a la base desde
                           articulo");}
//--------------   Seccion de ejecucion query   -------------------
  String res_cadena = articulobase.ConsultarTodosArticulos();
//--------------   Seccion de cierre de conexion   ----------------
    boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   -------------
    return(res_cadena);
}

public java.lang.String EliminarArticulo(int CodArticulo)
{//--------------   Seccion de apertura de conexion   -------------
   try{    //realiza una conexion a base
    if (articulobase.Conectar(url , usuario , clave)){
      System.out.println("Conectado para manejar articulo");}
    else{
      System.out.println("Error en la conexion a la base de
                          datos");
      return "-15";}
    }catch (Exception ee){
      System.out.println("Error al conectarse a la base desde
                          articulo");}
//--------------   Seccion de ejecucion query   -------------------
  String res_cadena = articulobase.EliminarArticulo(CodArticulo);
//--------------   Seccion de cierre de conexion   ----------------
  boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   ------------
  return(res_cadena);
}

public java.lang.String NuevoPrecioArticulo(int CodArticulo,long
                                             NuePrecio)
{//--------------   Seccion de apertura de conexion   -------------
   try{    //realiza una conexion a base
     if (articulobase.Conectar(url , usuario , clave)){
       System.out.println("Conectado para manejar articulo");}
     else{
       System.out.println("Error en la conexion a la base de datos");
       return "-15";}
```

```
   }catch (Exception ee){
    System.out.println("Error al conectarse a la base desde
                        articulo");}
//--------------   Seccion de ejecucion query   ------------------
    String res_cadena =
    articulobase.NuevoPrecioArticulo(CodArticulo,NuePrecio);
//--------------   Seccion de cierre de conexion   ----------------
    boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   --------------
    return(res_cadena);
}

public java.lang.String IncrementarStock(int CodArticulo,int
                            CantIncrementar)
{//--------------   Seccion de apertura de conexion   --------------
   try{    //realiza una conexion a base
     if (articulobase.Conectar(url , usuario , clave)){
       System.out.println("Conectado para manejar articulo");}
     else{
       System.out.println("Error en la conexion a la base de datos");
       return "-15";}
    }catch (Exception ee){
     System.out.println("Error al conectarse a la base desde
                         articulo");}
//--------------   Seccion de ejecucion query   -------------------
    String res_cadena =
    articulobase.IncrementarStock(CodArticulo,CantIncrementar);
//--------------   Seccion de cierre de conexion   ----------------
    boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   -------------
    return(res_cadena);
  }

public java.lang.String DevolucionArticulo(int CodArticulo,int
                                        CodFactura,int CanDevolver)
{//--------------   Seccion de apertura de conexion   --------------
 try{    //realiza una conexion a base
   if (articulobase.Conectar(url , usuario , clave)){
     System.out.println("Conectado para manejar articulo");}
   else{
     System.out.println("Error en la conexion a la base de datos");
     return "-15";}
  }catch (Exception ee){
   System.out.println("Error al conectarse a la base desde
                       articulo");}
//--------------   Seccion de ejecucion query   -------------------
   String res_cadena=articulobase.DevolucionArticulo(CodArticulo,
    CodFactura,CanDevolver);
 //------------   Seccion de cierre de conexion   -----------------
    boolean resp_cierre = CierraConexion();
 //--------------   Seccion de retorno de resultados   -------------
    return(res_cadena);
 }


 public java.lang.String AjusteArticulo(int CodArticulo,char
                                        MotAjuste,int  CanAjuste)
 {//--------------   Seccion de apertura de conexion   --------------
   try{    //realiza una conexion a base
     if (articulobase.Conectar(url , usuario , clave)){
        System.out.println("Conectado para manejar articulo");}
```

```
     else{
        System.out.println("Error en la conexion a la base de datos");
        return "-15";}
     }catch (Exception ee){
       System.out.println("Error al conectarse a la base desde
                           articulo");}
//--------------   Seccion de ejecucion query   --------------------
  String res_cadena = articulobase.AjusteArticulo(CodArticulo,
                                          MotAjuste,CanAjuste);
//--------------   Seccion de cierre de conexion   -----------------
  boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   ---------------
  return(res_cadena);
}

public java.lang.String ConsultarStock(int CodArticulo)
{//--------------   Seccion de apertura de conexion   -----------------
   try{    //realiza una conexion a base
     if (articulobase.Conectar(url , usuario , clave)){
         System.out.println("Conectado para manejar articulo");}
     else{
         System.out.println("Error en la conexion a la base de datos");
         return "-15";}
   }catch (Exception ee){
      System.out.println("Error al conectarse a la base desde articulo");}
//--------------   Seccion de ejecucion query   --------------------
  String res_cadena = articulobase.ConsultarStock(CodArticulo);
//--------------   Seccion de cierre de conexion   -----------------
    boolean resp_cierre = CierraConexion();
//--------------   Seccion de retorno de resultados   ---------------
    return(res_cadena);
   }
}
```

Código del Servidor

```
// ServidorVentas.java: Programa Principal del Servidor Ventas de Articulos

class ServidorVentas { static public void main(String[] args)
  { try
    { // Initiacializa el ORB
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
      // Initializa el BOA
      org.omg.CORBA.BOA boa = orb.BOA_init();
      // Crea los objetos : Articulo, Factura, Cliente, Vendedor
      ArticuloImpl articulo = new ArticuloImpl("Mi Articulo");
      FacturaImpl factura = new FacturaImpl("Mi Factura");
      ClienteImpl cliente = new ClienteImpl("Mi Cliente");
      VendedorImpl vendedor = new VendedorImpl("Mi Vendedor");
      // Exporta a el ORB los 4 objetos creados
      boa.obj_is_ready(articulo);
      boa.obj_is_ready(factura);
      boa.obj_is_ready(cliente);
      boa.obj_is_ready(vendedor);
      // Servidor listo para escuchar requerimientos
      boa.impl_is_ready();
      }
   catch(org.omg.CORBA.SystemException e)
      { System.err.println(e);  }
   }
}
```

Código del Cliente

El código del cliente sólo se presenta a una transacción que es CounsultarArtículo, únicamente para mostrar como es la invocación de los objetos.

```
import java.util.*;
import java.io.*;
import java.lang.*;
import java.awt.*;

public class ConsultarArticulo extends java.applet.Applet {
       //{{DECLARE_CONTROLS
       private java.awt.Label lbltitulo;
       private java.awt.Label lblcodigo;
       private java.awt.Label lbldescripcion;
       private java.awt.Label lblmarca;
       private java.awt.Label lblprecio;
       private java.awt.Label lbliva;
       private java.awt.TextField txtcodigo;
       private java.awt.TextField txtdescripcion;
       private java.awt.TextField txtmarca;
       private java.awt.TextField txtprecio;
       private java.awt.TextField txtiva;
       private java.awt.Button btnaceptar;
       //}}

      private Ventas.Articulo articulo;
      static String ArregloParametros[];

 public void init() {
       setLayout(null);
       setSize(363,200);
       lblcodigo = new java.awt.Label("Código :");
       lblcodigo.setBounds(60,48,48,24);
       lblcodigo.setFont(new Font("Dialog", Font.BOLD, 12));
       add(lblcodigo);
       lbldescripcion = new java.awt.Label("Descripción :");
       lbldescripcion.setBounds(60,84,72,24);
       lbldescripcion.setFont(new Font("Dialog", Font.BOLD, 12));
       add(lbldescripcion);
       lblmarca = new java.awt.Label("Marca :");
       lblmarca.setBounds(60,120,48,24);
       lblmarca.setFont(new Font("Dialog", Font.BOLD, 12));
       add(lblmarca);
       lblprecio = new java.awt.Label("Precio :");
       lblprecio.setBounds(60,156,48,24);
       lblprecio.setFont(new Font("Dialog", Font.BOLD, 12));
       add(lblprecio);
       lbliva = new java.awt.Label("% Iva :");
       lbliva.setBounds(60,192,48,24);
       lbliva.setFont(new Font("Dialog", Font.BOLD, 12));
       add(lbliva);
       txtcodigo = new java.awt.TextField();
       txtcodigo.setBounds(144,48,49,21);
       add(txtcodigo);
       txtdescripcion = new java.awt.TextField();
       txtdescripcion.setEditable(false);
       txtdescripcion.setBounds(144,84,250,21);
```

```
        txtdescripcion.setBackground(java.awt.Color.lightGray);
        add(txtdescripcion);
        txtmarca = new java.awt.TextField();
        txtmarca.setEditable(false);
        txtmarca.setBounds(144,120,100,21);
        txtmarca.setBackground(java.awt.Color.lightGray);
        add(txtmarca);
        txtprecio = new java.awt.TextField();
        txtprecio.setEditable(false);
        txtprecio.setBounds(144,156,100,21);
        txtprecio.setBackground(java.awt.Color.lightGray);
        add(txtprecio);
        txtiva = new java.awt.TextField();
        txtiva.setEditable(false);
        txtiva.setBounds(144,192,49,21);
        txtiva.setBackground(java.awt.Color.lightGray);
        add(txtiva);
        btnaceptar = new java.awt.Button();
        btnaceptar.setLabel("Aceptar");
        btnaceptar.setBounds(150,240,60,26);
        btnaceptar.setBackground(java.awt.Color.lightGray);
        add(btnaceptar);
   try
     { // Initialize the ORB
       showStatus("INICIALIZANDO EL ORB");
       org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(this, null);
       // Bind to the Articulo Object
       showStatus("ENCONTRANDO EL OBJETO ARTICULO");
       articulo = Ventas.ArticuloHelper.bind(orb, "Mi Articulo");
    }//fin try
    catch(org.omg.CORBA.SystemException e)
{   showStatus("Applet Exception" + e); }
}//fin init

public boolean action(Event ev,Object arg) {
    int codigo =0;
    if(ev.target == btnaceptar) {
      if ( !((txtcodigo.getText()).trim()).equals("") ) {
       try{
          codigo =  Integer.parseInt((txtcodigo.getText()).trim());
       }catch(NumberFormatException e)
       {  showStatus("ERROR,CODIGO NO VALIDO");
          txtcodigo.setText("");
          txtdescripcion.setText("");
          txtmarca.setText("");
          txtprecio.setText("");
          txtiva.setText("");
          return false;
       }
       try
       {
          String contenido="";
          showStatus("EJECUTANDO LA CONSULTA DEL ARTICULO:" + codigo);
          contenido=articulo.ConsultarArticulo(codigo);
          int ii = SeparaParametros(contenido);
          if (ArregloParametros.length > 0 ) {
              if (ArregloParametros.length > 1 )  {
                 if ( ArregloParametros[0].equals("1") )  {
                    txtdescripcion.setText(ArregloParametros[2]);
                    txtmarca.setText(ArregloParametros[3]);
                    txtprecio.setText(ArregloParametros[4]);
```

```
                    txtiva.setText(ArregloParametros[5]);
                    showStatus("SU TRANSACCION SE REALIZO CON EXITO");
                    return true;
                  }
                  else //1 {
                 showStatus("Error Servidor Retorno :" + ArregloParametros[0]);
                 return false;
                  }
                 }
                else  //2 {
                 showStatus("Error Servidor Retorno :" + ArregloParametros[0]);
                  return false;
                 }
              }
              else //3 {
                      showStatus("SERVIDOR NO RETORNO NADA");
                      return false;
               }
            }//fin try
               catch(org.omg.CORBA.SystemException e) {
               showStatus("System Exception " + e);
               return false;
            }
         }
         else {
            showStatus("ERROR, CODIGO NO VALIDO");
            txtcodigo.setText("");
            txtdescripcion.setText("");
            txtmarca.setText("");
            txtprecio.setText("");
            txtiva.setText("");
            return false;
      }
   }//fin if
   return false;
}//fin action

private static int SeparaParametros(String Parametros) {
    int i;
    StringTokenizer st;
    // Creamos el objeto StringTokenizer
    st = new StringTokenizer( Parametros,";" );
    // Creamos el almacenamient
    ArregloParametros= new String[ st.countTokens() ];
    // Separamos los tokens de la cadena del parámetro
    for( i=0; i < ArregloParametros.length; i++ )
              ArregloParametros[i] = st.nextToken();
              return 1;
 }//fin SeparaParametros

}//fin class ConsultarArticulo
```

Anexo 3

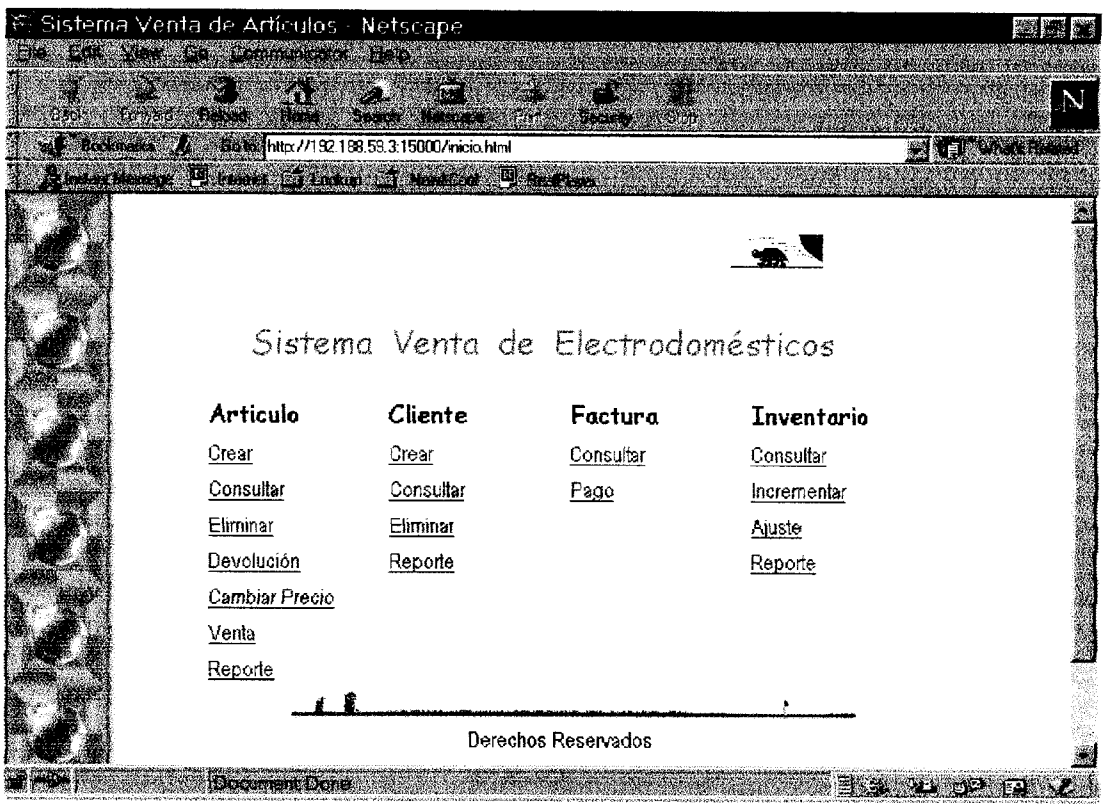
Manual del Usuario

Conectándose al Sistema Venta de Electrodomésticos

Para conectarse al Sistema Venta de Electrodomésticos, escriba en la línea "location" ó "localización" del browser lo siguiente:

http://dirección-IP-webserver : 15000/inicio.html

Donde dirección-IP-webserver.- Corresponde a la dirección del computador donde está levantado el WebServer y 15000 es el puerto en que está escuchando el WebServer.



Cuando obtenemos la página inicio.html del WebServer, tenemos el menú principal que contiene 4 menús:

1. Artículo
2. Cliente
3. Factura
4. Inventario

En el menú artículo, tiene las siguientes opciones:

- Crear
- Eliminar
- Consultar
- Devolución
- Cambiar Precio
- Venta
- Reporte

En el menú cliente, tiene las siguientes opciones:

- Crear
- Eliminar
- Consultar
- Reporte

En el menú factura, tiene las siguientes opciones:

- Consultar
- Pago

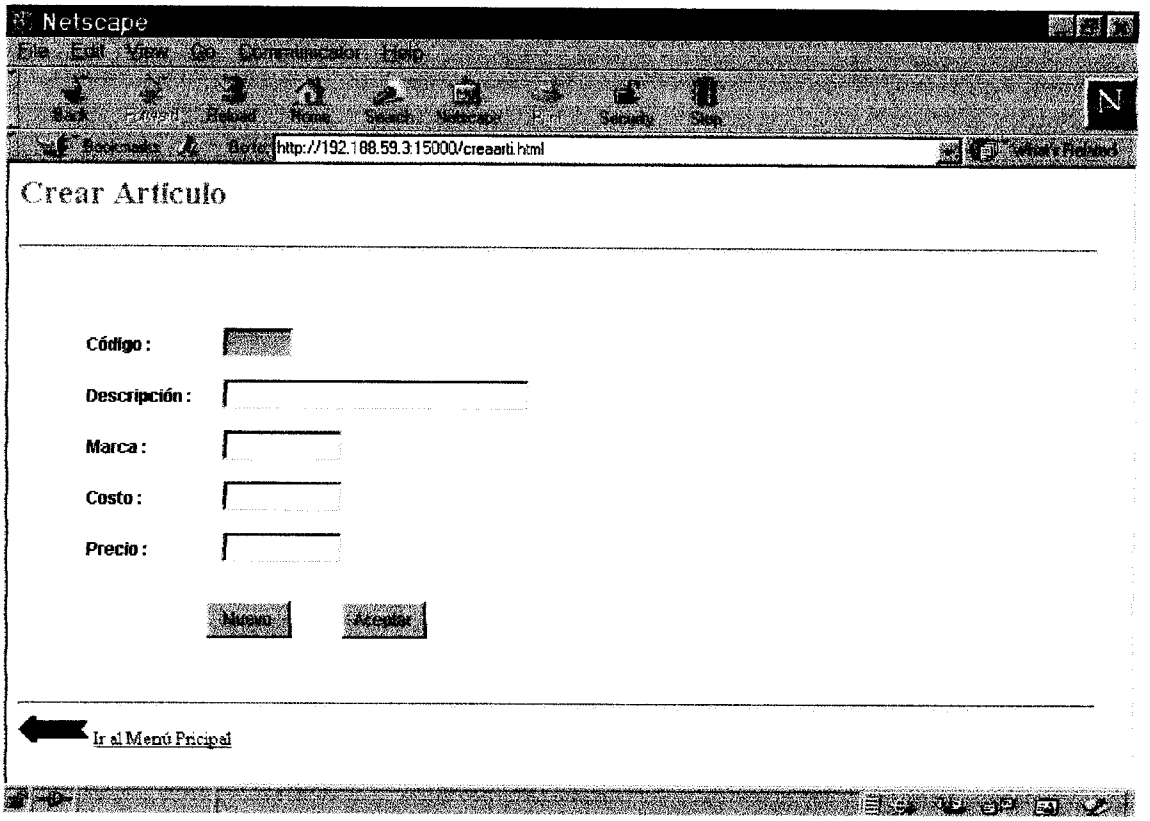
En el menú inventario, tiene las siguientes opciones:

- Consultar
- Incrementar
- Ajuste
- Reporte

A continuación se explicaran con más detalles lo que significa cada opción y como debemos utilizarlas.

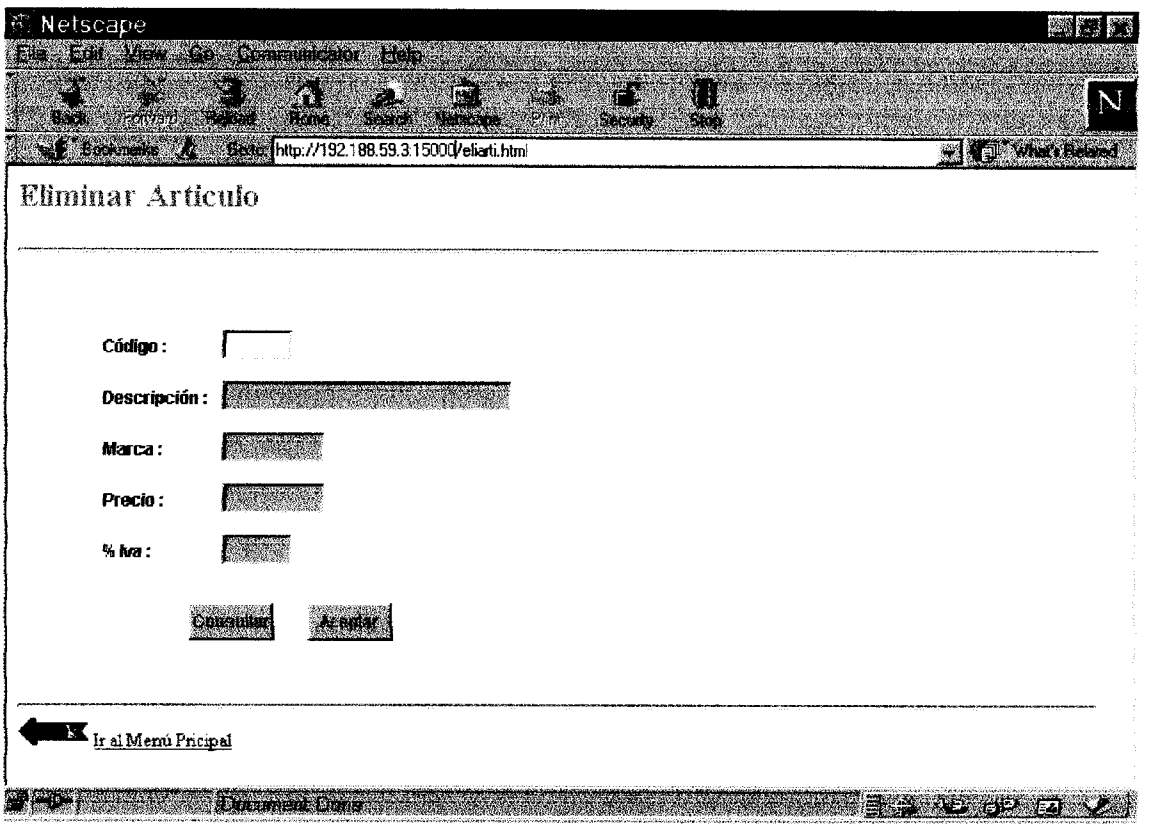
Opción Crear Artículo

Esta opción permite ingresar un nuevo artículo al sistema, para hacerlo haga un click en el botón Nuevo para tomar el siguiente código disponible en la base; a continuación entre los respectivos detalles del artículo y cuando finalice haga un click en el botón Aceptar para guardar los datos en la base.



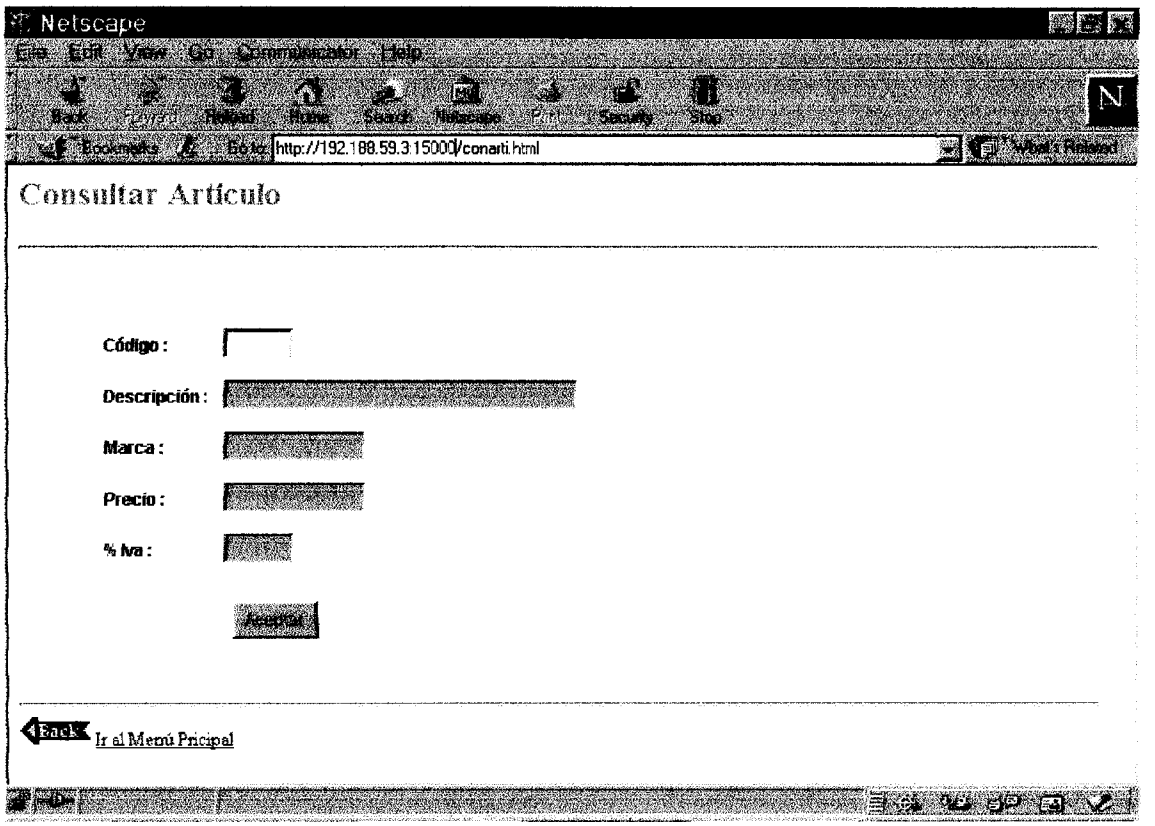
Opción Eliminar Artículo

Esta opción permite eliminar artículo del sistema, escriba el código del artículo a eliminar y haga un click en el botón *Consultar* para verificar si es el artículo correcto. Ahora para eliminar el artículo haga click en el botón *Aceptar*, si hay stock del artículo y/o deberá primero ajustar a la cantidad a 0 entrando en la opción Ajuste del menú Inventario.



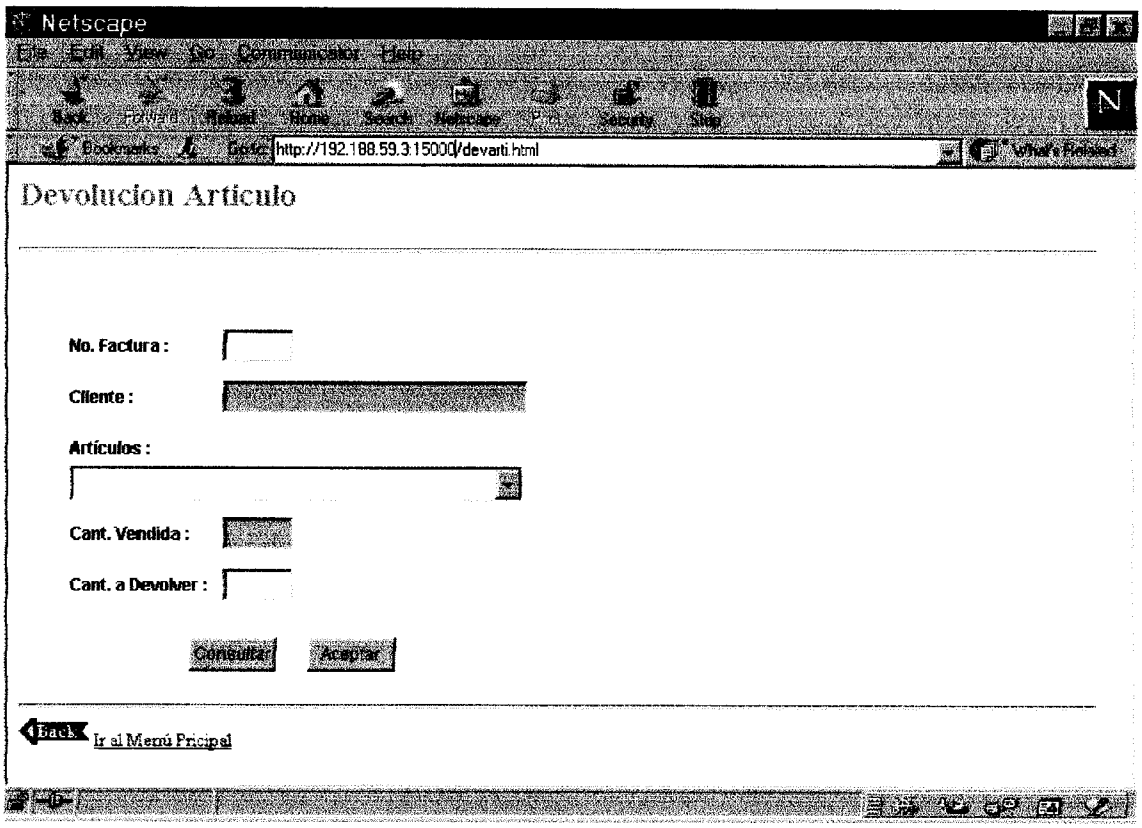
Opción Consultar Artículo

Esta opción permite consultar un artículo para revisar sus detalles, escriba el código del artículo y haga un click en el botón *Aceptar.*



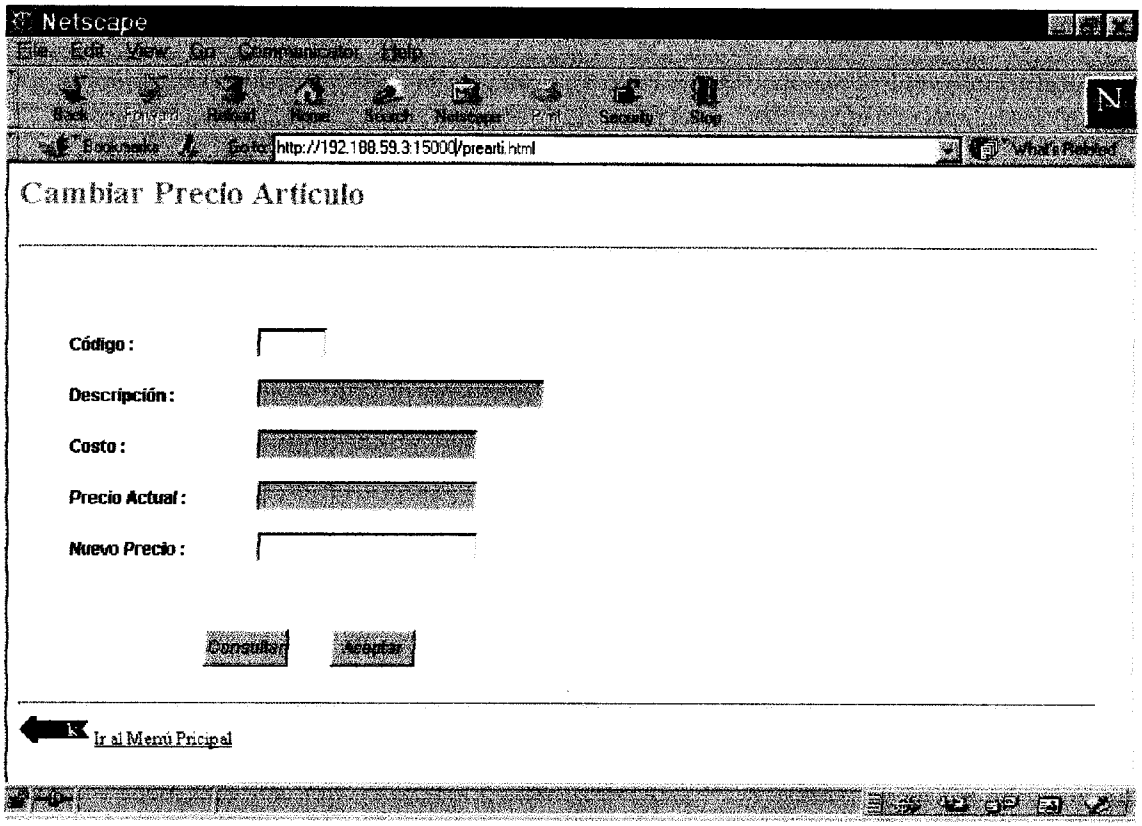
Opción Devolución de Artículo

Esta opción permite realizar una devolución de un artículo que ya fue previamente vendido, escriba el número de factura y haga un click en el botón *Consultar* para verificar si es la factura correcta, luego seleccione el artículo a devolver en la lista donde aparece el detalle de la factura y después escriba la cantidad a devolver, finalmente haga un click en el botón *Aceptar*.



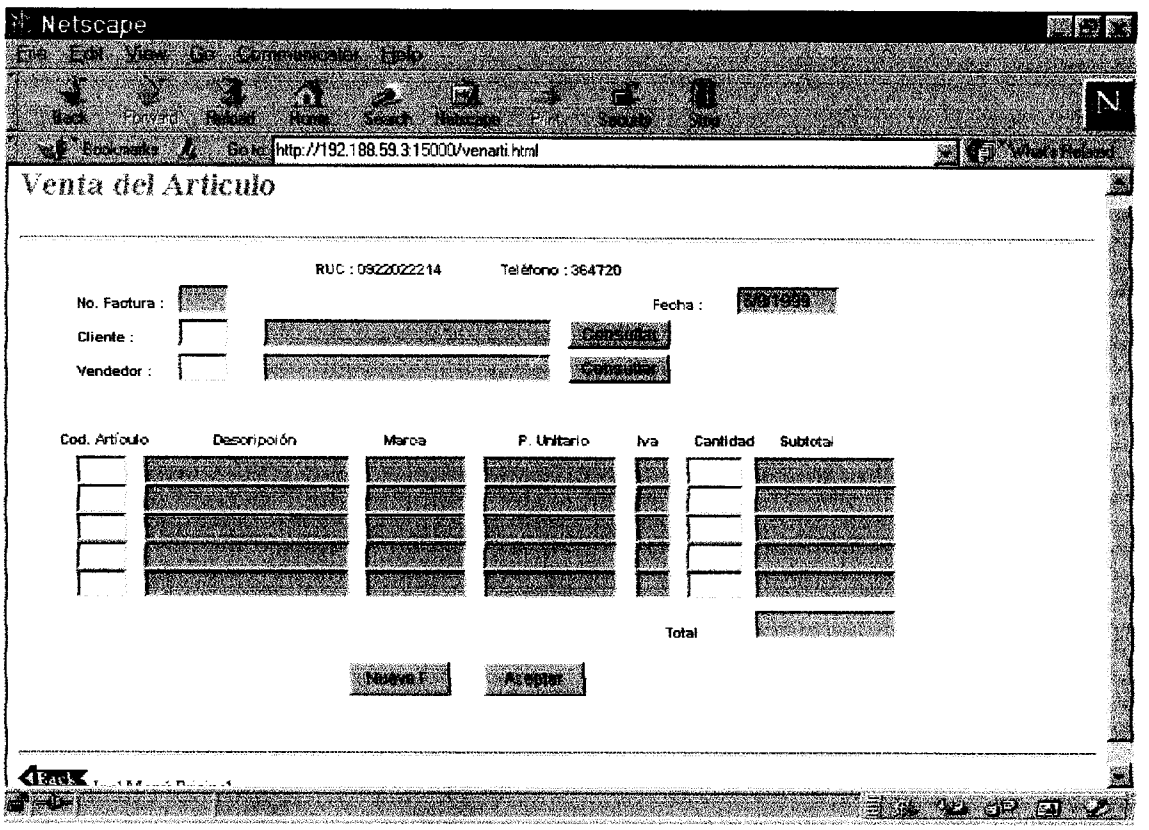
Opción Cambiar Precio del Artículo

Esta opción permite cambiar el precio de venta de un artículo, escriba el código del artículo y haga un click en el botón *Consultar* para verificar si se trata del artículo correcto, luego escriba el nuevo precio de venta y finalmente haga click en el botón *Aceptar* para guardar los cambios en la base.



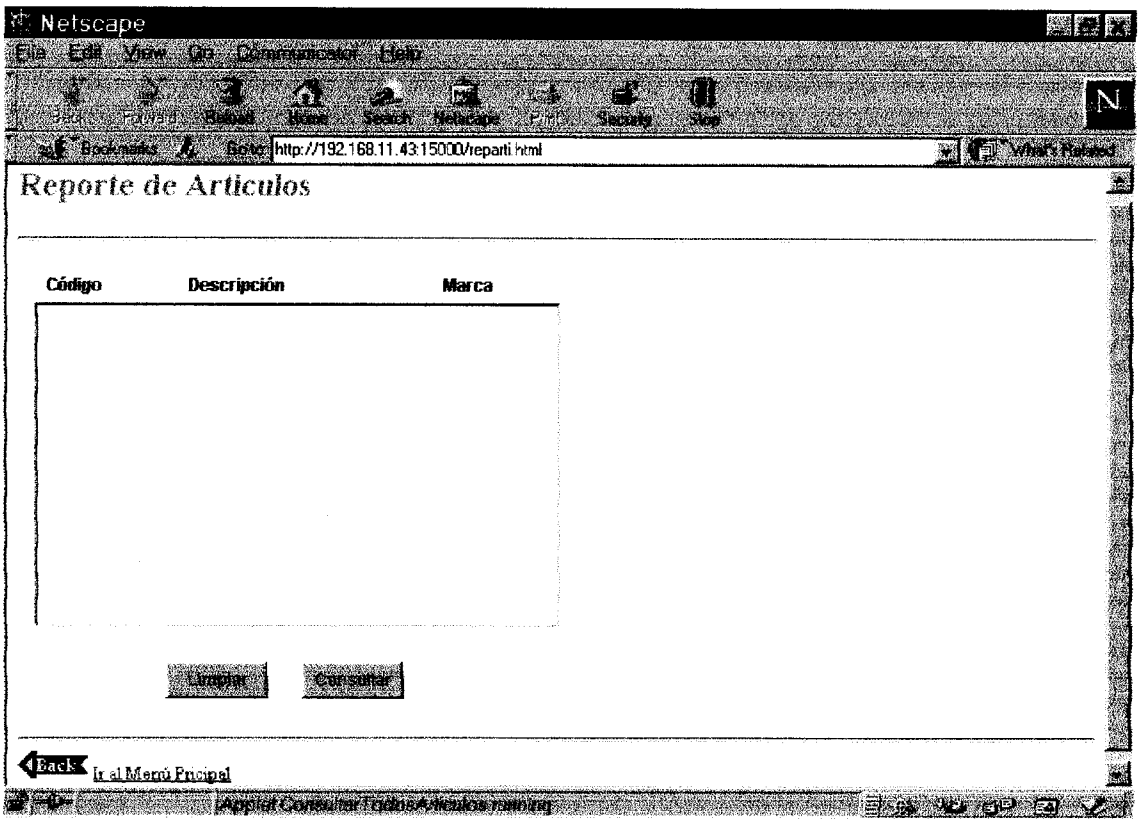
Opción Venta del Artículo

Esta opción permite realizar la venta de los artículos, haga un click en el botón *Nuevo* para generar el siguiente número de factura disponible. Escriba después el código del cliente y haga click en el botón *Consultar* para traer los datos del cliente, luego escriba el código del vendedor y haga click en el botón *Consultar* para traer los datos del vendedor. Para realizar la venta de los artículos debe moverse en el detalle con la tecla TAB a medida que va escribiendo el código del artículo y la cantidad a vender. Finalmente para guardar la factura en la base haga un click en el botón *Aceptar.*



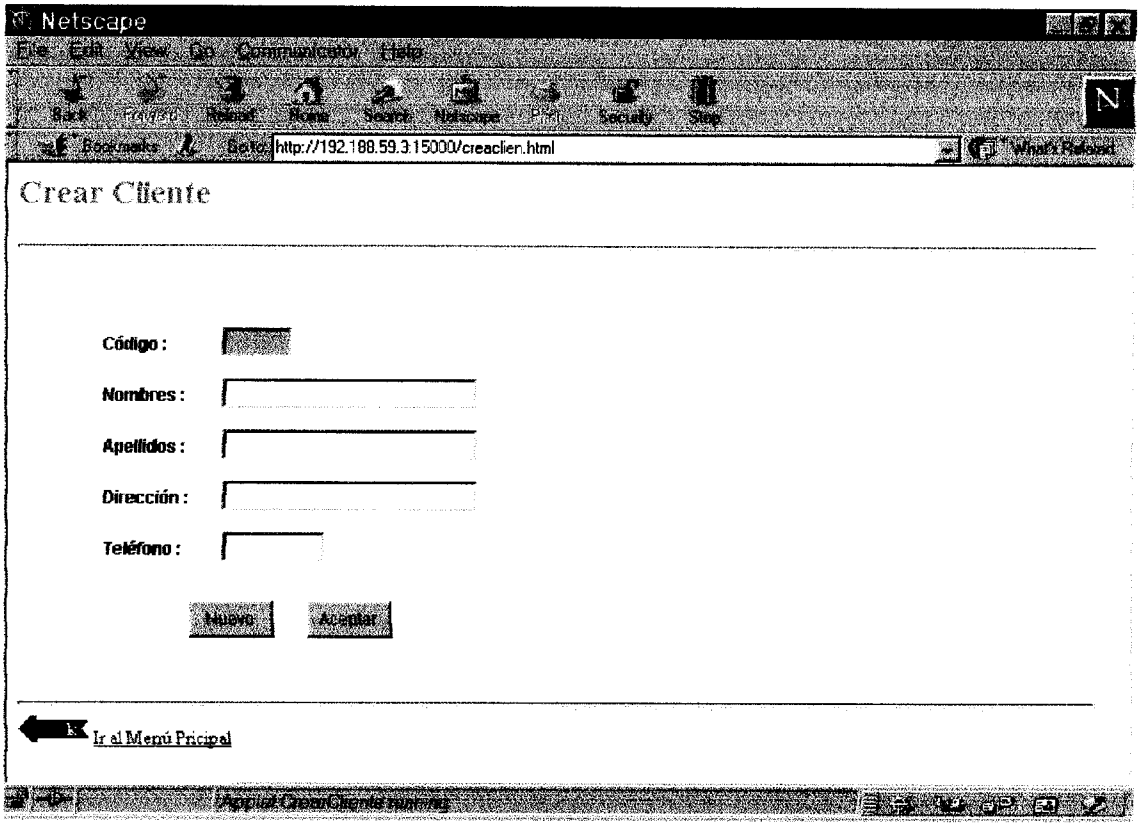
Opción Reporte de Artículos

Esta opción permite realizar un reporte o listado de todos los artículos que existen en la base de datos. Haga un click en el botón *Consultar.*



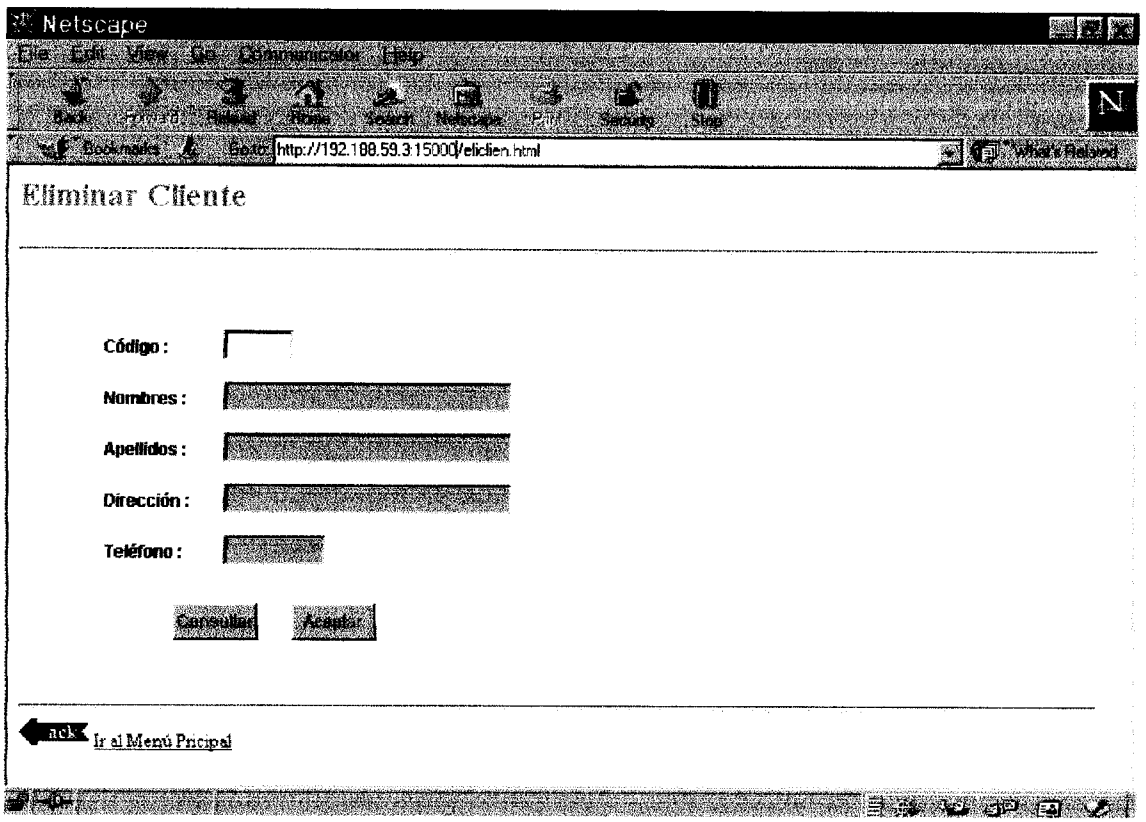
Opción Crear Cliente

Esta opción permite ingresar un nuevo cliente al sistema, para hacerlo haga un click en el botón Nuevo para tomar el siguiente código disponible en la base; a continuación entre los respectivos datos del cliente y cuando finalice haga un click en el botón Aceptar para guardar los datos en la base



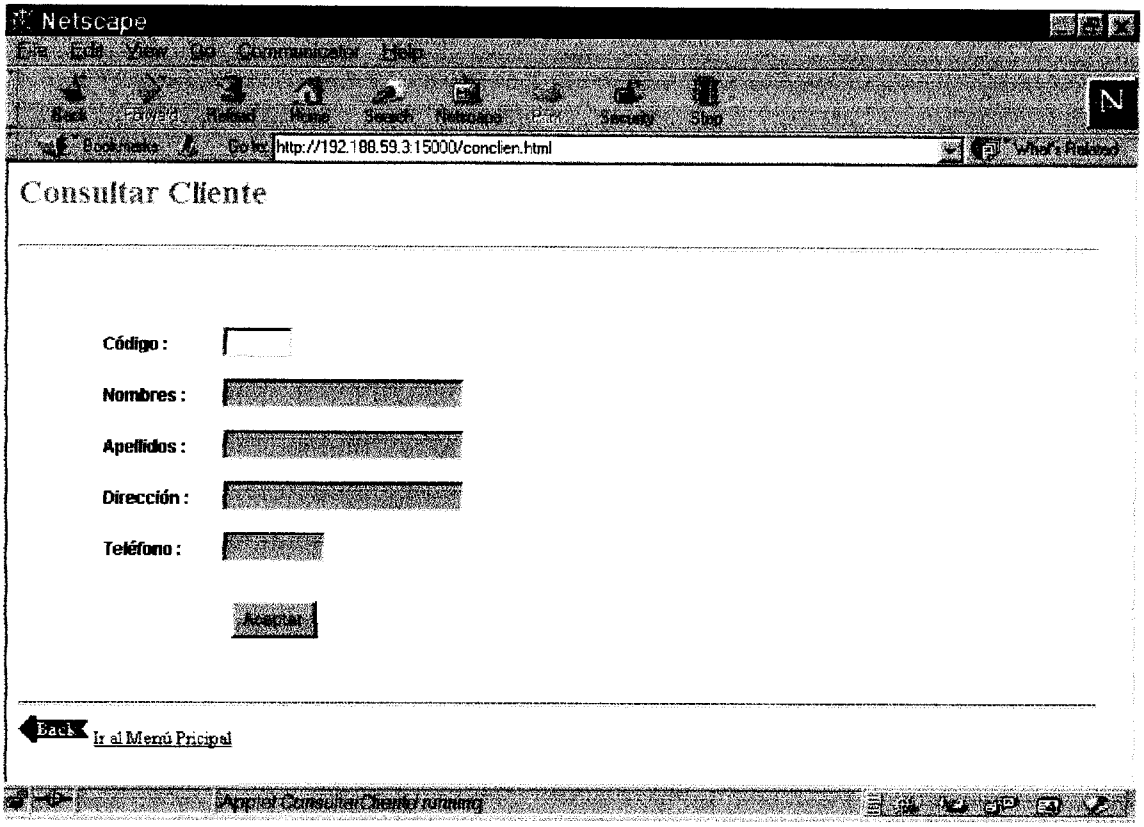
Opción Eliminar Cliente

Esta opción permite eliminar un cliente del sistema, escriba el código del cliente a eliminar y haga un click en el botón *Consultar* para verificar si es el cliente correcto. Finalmente para eliminar el cliente haga click en el botón *Aceptar.*



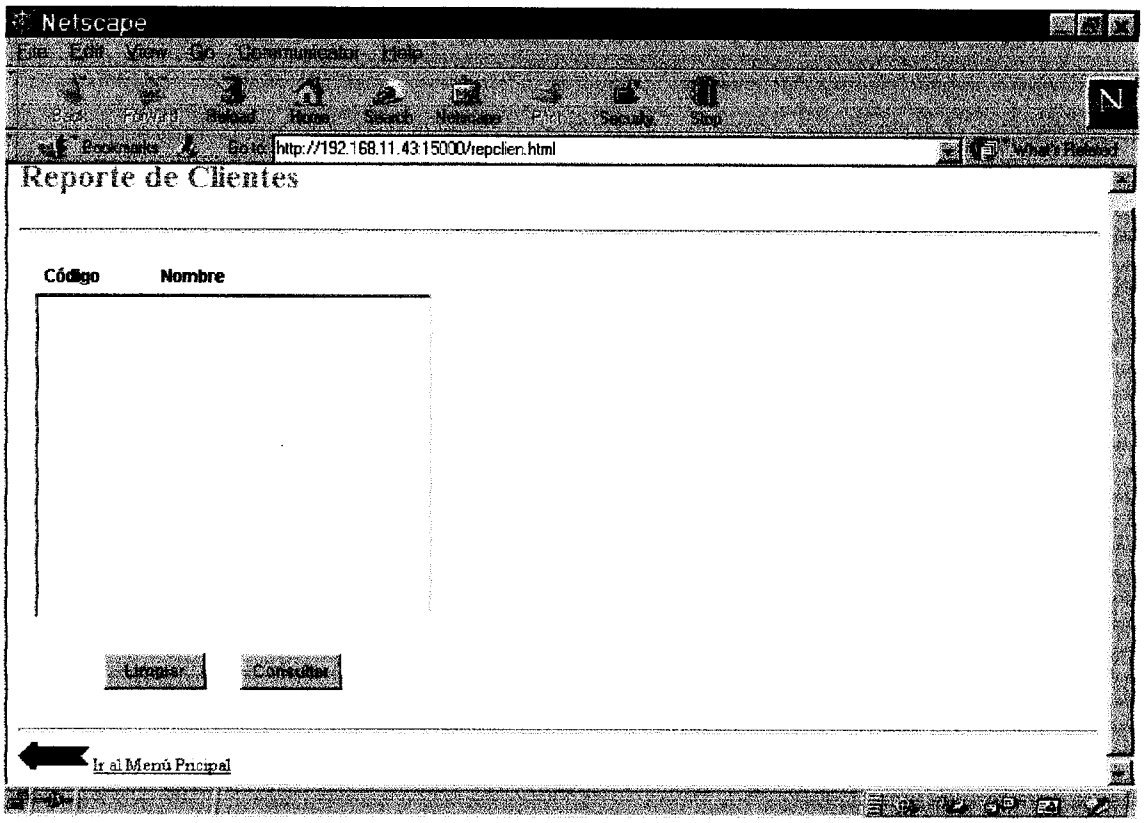
Opción Consultar Cliente

Esta opción permite consultar un cliente para revisar sus datos, escriba el código del cliente y haga un click en el botón *Aceptar*.



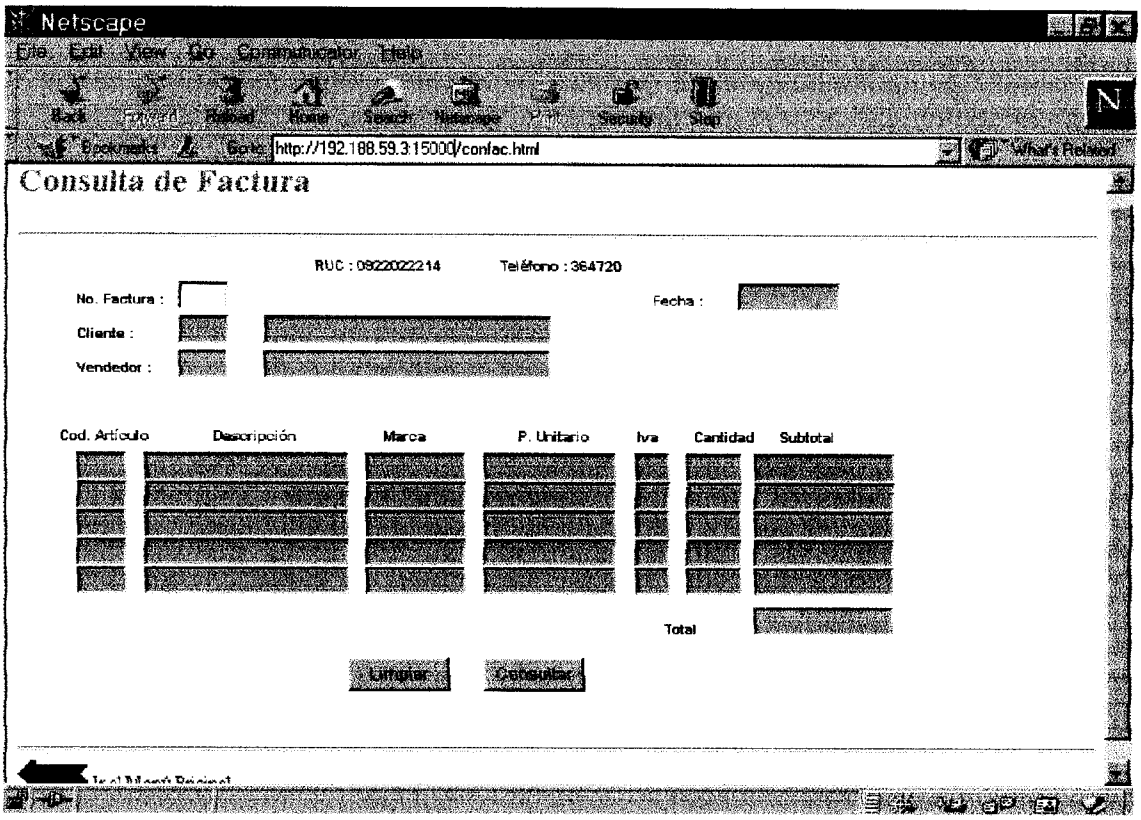
Opción Reporte de Clientes

Esta opción permite realizar un reporte o listado de todos los clientes que existen en la base de datos. Haga un click en el botón *Consultar*.



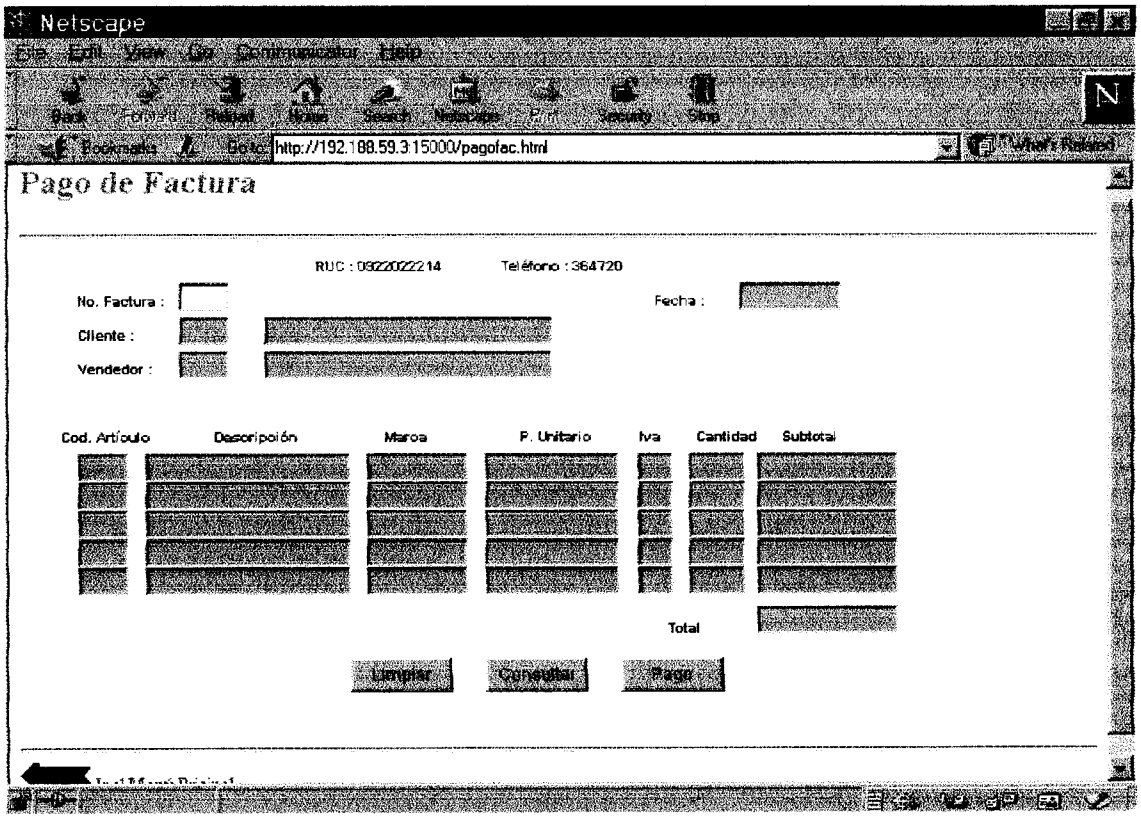
Opción Consultar Factura

Esta opción permite consultar una factura para revisar sus detalles, escriba el número de la factura y haga un click en el botón *Aceptar.*



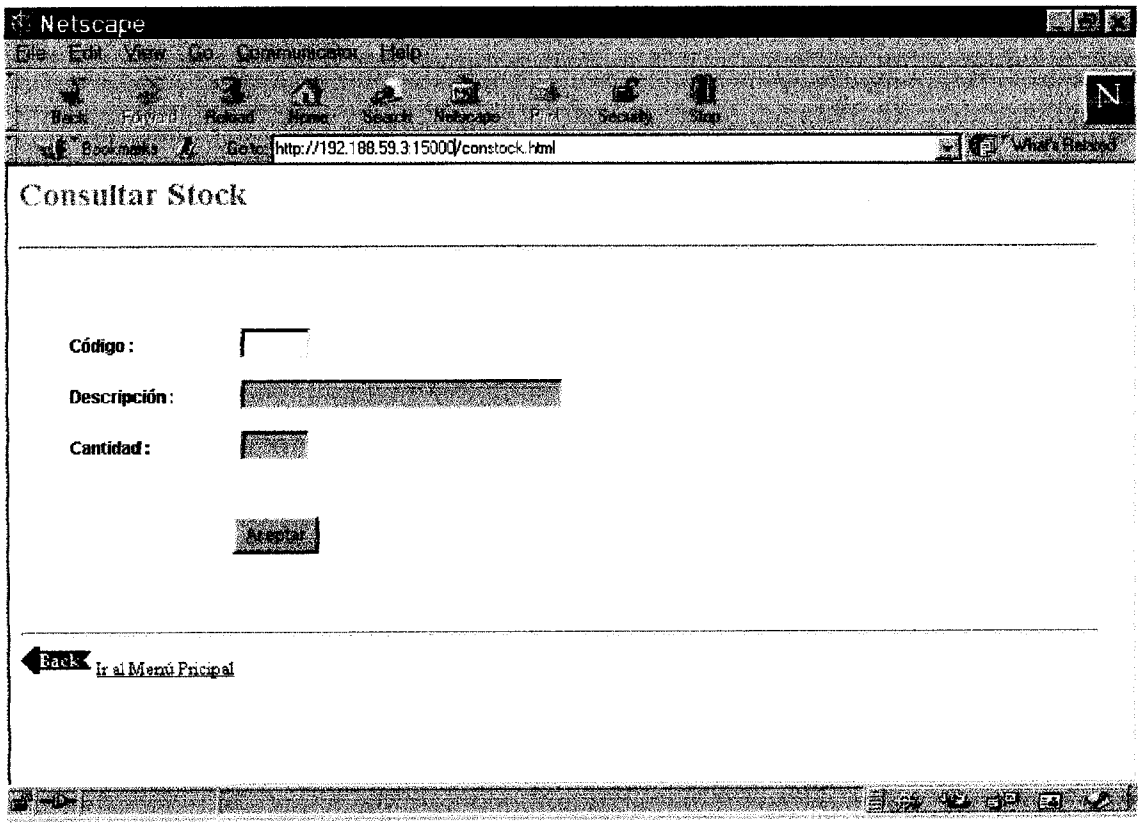
Opción Pago de Factura

Esta opción permite realizar el pago de una factura, escriba el número de la factura y haga un click en el botón *Consultar* para verificar si se trata de la factura correcta. Finalmente haga un click en el botón *Aceptar* para registrar el pago de la misma.



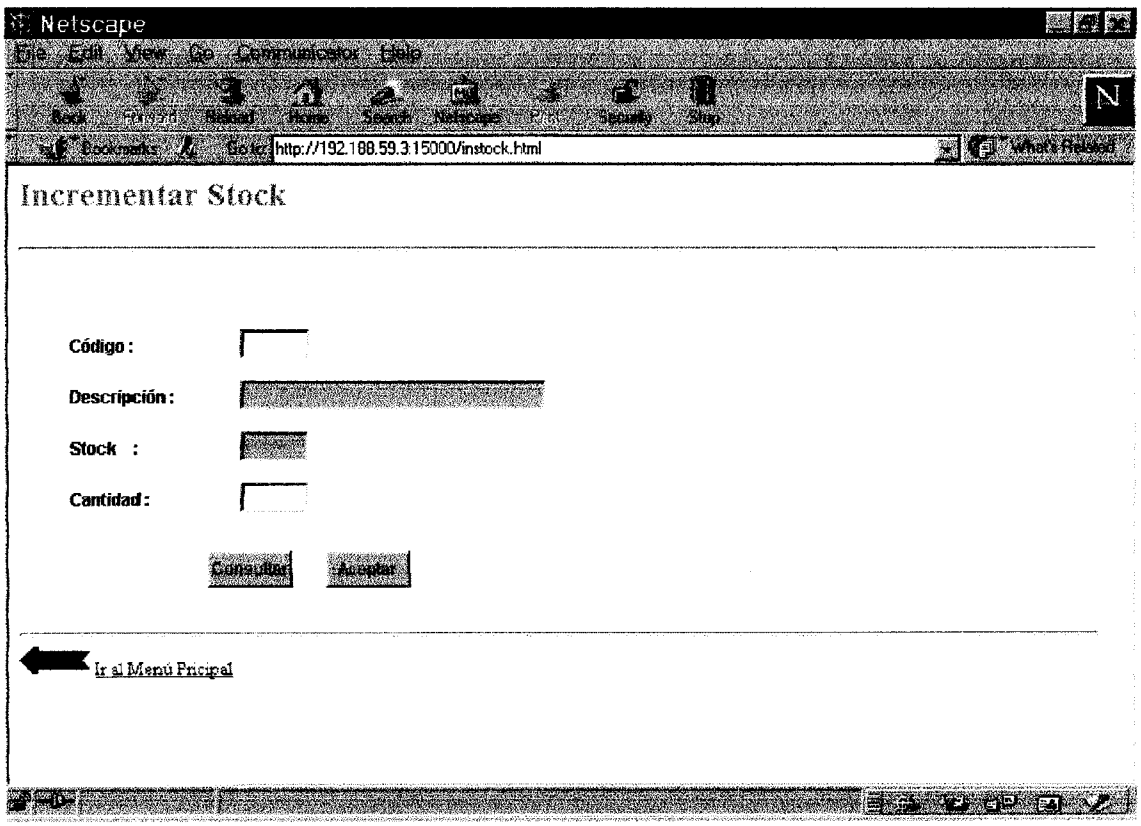
Opción Consultar Stock

Esta opción permite consultar el stock de un artículo, escriba el código del artículo y haga un click en el botón *Aceptar*



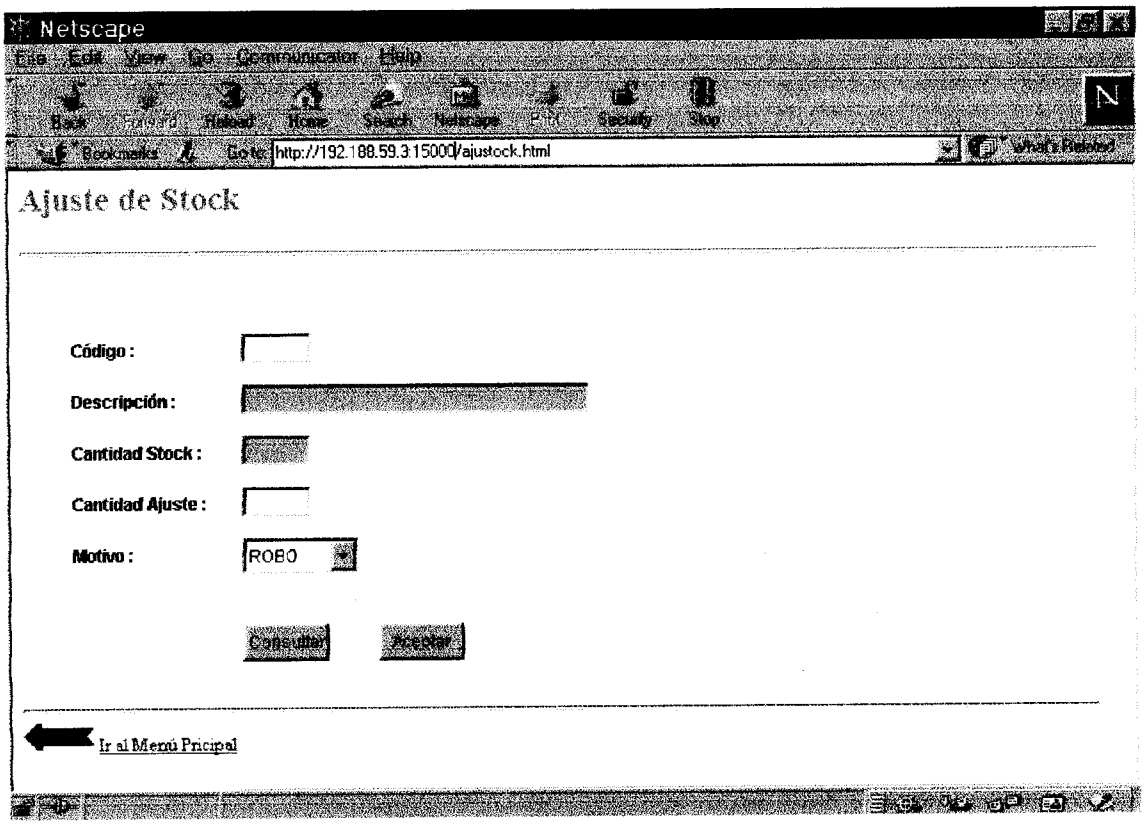
Opción Incrementar Stock

Esta opción permite incrementar el stock de un artículo, escriba el código y haga un click en el botón *Consultar* para verificar si se trata del artículo correcto. Escriba después la cantidad a incrementar y haga un click en el botón *Aceptar* para guardar los cambios en la base.



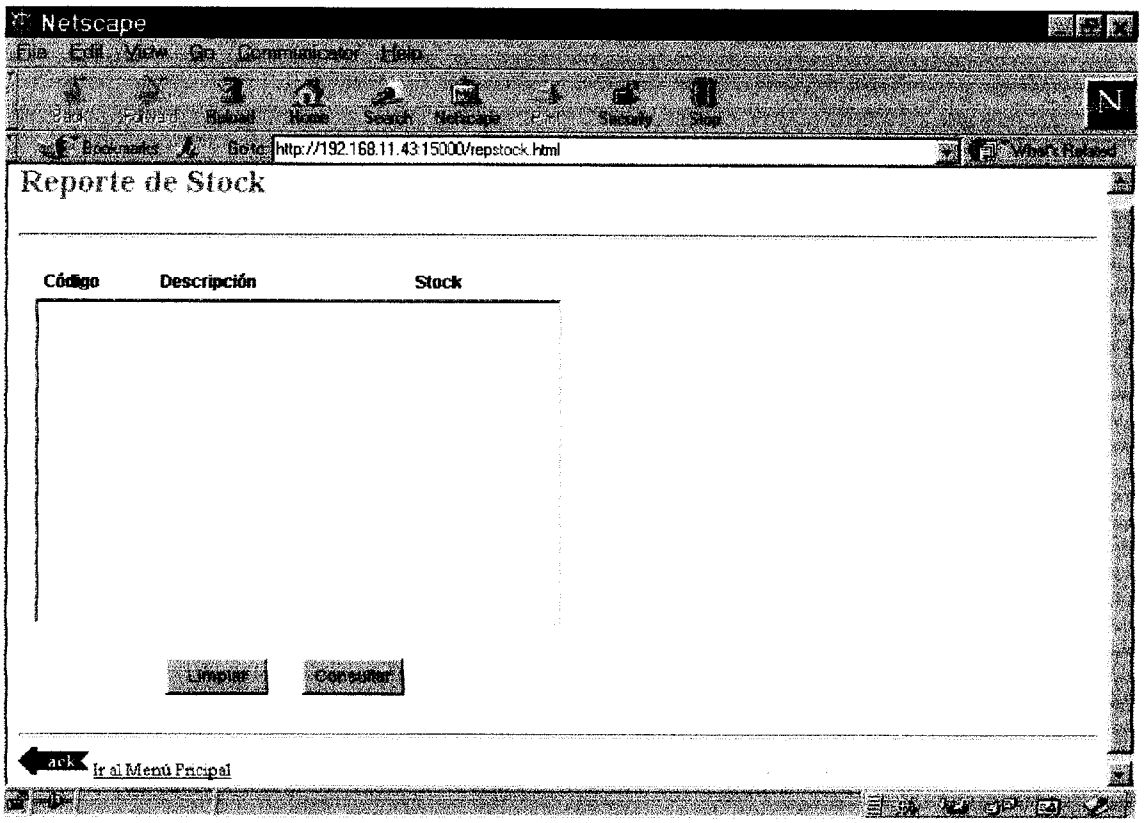
Opción Ajuste de Stock

Esta opción permite ajustar las cantidades de los artículos en el stock, escriba el código y haga un click en el botón *Consultar* para verificar si se trata del artículo correcto. Escriba la cantidad a ajustar y seleccione de la lista el motivo del ajuste (robo, daño ó baja). Finalmente haga un click en el botón *Aceptar* para guardar los cambios en la base.



Opción Reporte de Stock de Todos los Artículos

Esta opción permite realizar un reporte o listado del stock de todos los artículos que existen en la base de datos. Haga un click en el botón *Consultar*.



Anexo 3

Manual del Administrador

Para instalar el Sistema de Venta de Electrodomésticos, se debe hacer lo siguiente:

Configurando el ODBC para acceso de la base desde el Servidor Corba al Administrador de BD.- En Windows haga un click en el menú inicio, escoja del menú la opción Configuración luego seleccione Panel de Control y en esta ventana haga 2 clicks en el icono ODBC. Seleccione la pestaña System DNS, haga un click en el botón Add y escoja SQL Server debe poner entonces:

Nombre de la BD: artículos

User: alex

Password: alex

Instalando el Servidor Corba.- En el disco duro cree una carpeta con el nombre **C:\Servidor,** copie todos los archivos.class y el paquete *Ventas* que corresponde al servidor.

Levantando el Servidor Corba.- Habrá una ventana de DOS y entre a la carpeta servidor y escriba la siguiente línea:

C:\Servidor> start vbj ServidorVenta

Instalando el WebServer.- En la herramienta Visibroker viene incluido un servicio que se llama *gatekeeper*, éste actúa como protocolo IIOP y como WebServer. Cree una carpeta en el disco duro con el nombre **C:\WebServer**, copie todas las páginas html, los applets .class, los archivos *.jar de Visibroker y el paquete *Ventas*.

Levantando el WebServer.- Antes de levantar el WebServer, deberá primero descomprimir los archivos *.jar, vaya a una ventana de DOS entre a la carpeta WebServer y escriba la siguiente línea:

C:\WebServer> jar –xf archivo.jar

Después de descomprimir cada una de las librerías de Visibroker (*.jar), levante el servicio gatekeeper escribiendo la siguiente línea:

C:\WebServer> gatekeeper

Levantando el Cliente.- El cliente no necesita instalación alguna, sólo debe de levantarse el browser Nestcape Communicator y en la línea de localización escriba lo siguiente:

http://dirección-IP-webserver : 15000/inicio.html

GLOSARIO

API	Application Programming Interface
BD	Base de Datos
CORBA	Common Object Request Broker Architecture
DBMS	Database Management System
GCI	Gateway Commom Interface
GUI	Graphics Usuary Interface
HTTP	Hypertext Transfers Protocols
IDL	Interface Definition Language
IIOP	Internet Inter ORB Protocol
IP	Internet Protocol
JDBC	Java Database Connectivity
NOS	Network Operative System
ODBC	Open Database Connectivity
OA	Object Adapter
OI	Object Implement
OMG	Object Management Group
ORB	Object Request Broker
RPC	Remote Procedure Call
SQL	Structured Query Languaje
TCP	Transmission Control Protocol

BIBLIOGRAFIA

1. ORFALI, Robert. *The Essential Client /Server Survival Guide,* 2da Edición, Prentice Hall, 1996, 644 p.
2. ORFALI, Robert. *Client /Server Programming with Java and CORBA,* 2da Edición, Prentice Hall, 1998, 973 p.
3. MOWBRAY, Thomas. *The Essential CORBA: Systems Integration using Distributed Objects,* 1995.
4. COMER, Douglas. *Redes Globales de Información con Internet y TCP/IP. Principios básicos, protocolos y arquitectura,* 3ra Edición, Prentice Hall, 1996, 597 p.
5. LEWIS, Geoffrey. *Programming with Java IDL,* 1998.
6. VOGEL, Andreas. *Java Programming with CORBA,* 1997.