UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"SISTEMA DE VENTA EN LÍNEA PARA: CONSUMIBLES DE COMPUTACIÓN Y PAPELERÍA ÁNGELES"

T R A B A J O E S C R I T O
EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A:

R O X A N A A L E J A N D R A
V A R G A S C O N T R E R A S

FES Aragón

ASESOR: M. EN C. MARCELO PÉREZ MEDEL

MÉXICO, 2005.

ÍNDICE

INTRODUCCIÓN

El presente trabajo trata de los temas expuestos en el "2° *Diplomado de Desarrollo de Sistemas en Web*" que fue realizado del 05/Septiembre/2004 al 20/Abril/2005 y cuyo Coordinador Académico fue el Ing. Víctor Ramón Aguilar Ocampo. Cada capítulo corresponde a un módulo de dicho diplomado y de lo expuesto ahí, corresponde a la parte teórica del proyecto. La segunda parte de cada capítulo corresponde a la aplicación de dichos conocimientos al proyecto, en algunos casos no existe documentación, ya que el proyecto quedó en forma teórica y la parte de aplicación no se pudo realizar por falta de tiempo.

En el primer capítulo se verán los conceptos básicos necesarios para el desarrollo de un proyecto orientado a Web, entre ellos se encuentra que es Internet, sus servicios y las herramientas esenciales para el desarrollo de un proyecto, como son en nuestro caso Tomcat y MySQL. La segunda parte contiene la documentación de la propuesta del proyecto, con la solución a plantear, sus objetivos, riesgos, beneficios y alcance.

A continuación, en el segundo capítulo se tratan los elementos para el análisis y diseño del proyecto, utilizando herramientas UML, en nuestro caso Poseidón y las recomendaciones para realizar casos de uso correctos. Aquí se añaden algunos diagramas UML y la documentación de casos de uso, así como los análisis de requerimientos funcionales y no funcionales del proyecto.

Después, en nuestro tercer capítulo se presenta la sintaxis básica del Lenguaje Java, y sus tipos de aplicaciones, dado que este será el lenguaje utilizado para desarrollar la aplicación, puesto que cuenta con numerosos módulos que facilitan la orientación a Web, como son los API´s JDBC, Struts e Ibatis entre otros. La parte que corresponde a la aplicación de estos conocimientos en el proyecto no se realizó por falta de tiempo, pero se anexan programas básicos que sirven para la administración y la creación de tablas en la base de datos, los cuales son la base para construir la aplicación.

Más tarde, en el cuarto capítulo se muestran normas y recomendaciones para el diseño de la interfaz de usuario, es decir, la parte del sistema que permitirá el acceso a la aplicación, esto se realizará por medio de páginas HTML, Dreamweaver será la herramienta de desarrollo utilizada para la creación de la

interfaz, además se verá el lenguaje HTML y sus tags o etiquetas más comunes. En éste capítulo se anexan las pantallas principales del sistema, así como la codificación HTML de la página principal, así como las especificaciones de la Base de Datos del proyecto.

Luego, en el quinto capítulo se verán las reglas y recomendaciones para diseñar y construir la base de datos del proyecto, para ello se utilizará el manejador de bases de datos MySQL, y el lenguaje SQL para la creación de las tablas y las consultas, también se verá la sintaxis para crear la base de datos y realizar consultas a ésta. En ésta parte se añade el Diagrama Entidad-Relación de la Base de Datos del proyecto, junto con su diccionario de datos.

Por último, en el sexto capítulo se muestra como aplicar las reglas del negocio en nuestra aplicación, utilizando los API's Struts e Ibatis de Java, los cuales permiten la implementación de las reglas del negocio en nuestra aplicación y dan a su vez integridad y seguridad a los datos que se almacenarán en nuestra base de datos. En éste capítulo no se añade documentación respecto al proyecto, puesto que la parte de implementación de reglas del negocio del proyecto utilizando Ibatis y Struts no se pudo realizar por falta de tiempo.

CAPÍTULO 1. CONCEPTOS BÁSICOS PARA EL DISEÑO DE UN PROYECTO ORIENTADO A WEB

1.1 INTRODUCCIÓN A INTERNET

1.1.1 ¿Qué es Internet?

Internet es una red de redes a escala mundial de millones de computadoras interconectadas con el conjunto de protocolos TCP/IP[1], es decir, es una red mundial de equipos de cómputo que se comunican mediante un lenguaje común conocido como protocolo de red. El protocolo de red utilizado en Internet es el TCP/IP. Cabe aclarar que Internet no es sinónimo de World Wide Web, ya que la World Wide Web es uno de los muchos servicios ofertados en la red Internet.

1.1.2 Orígenes de Internet

A finales de 1972 apareció ARPANET[2], una nueva red de comunicaciones financiada por la DARPA que funcionaba sobre la red telefónica conmutada. En 1973, la DARPA inició un programa de investigación sobre posibles técnicas para interconectar redes (orientadas al tráfico de paquetes) de distintas clases, desarrollando nuevos protocolos de comunicaciones que permitían intercambio de información de forma "transparente" para los ordenadores conectados. De la filosofía del proyecto surgió el nombre de "Internet", que se aplicó al sistema de redes interconectadas mediante los protocolos TCP e IP.

El 1 de enero de 1983 ARPANET cambió el protocolo NCP[3] por TCP/IP. Ese mismo año, se creó el IAB[4] con el fin de estandarizar el protocolo TCP/IP y de proporcionar recursos de investigación a Internet.

En 1986 la NSF[5] comenzó el desarrollo de NSFNET que se convirtió en la principal red troncal de Internet, complementada después con las redes NSINET y ESNET,

---

[1] Protocolo creado para Internet, unión de los protocolos de red TCP e IP.

[2] DARPA (Defense Advanced Research Projects Agency) agencia gubernamental de investigación de proyectos avanzados en Defensa, creada en respuesta a los desafíos tecnológicos y militares de Rusia de la cual surgirán los fundamentos de la futura red global de computadoras el Internet.

[3] NCP (Network Core Protocol), NCP es un protocolo de red orientado a conexión, en el cual se basa TCP.

[4] Fundación Nacional de Ciencias acrónimo en inglés, NSF (National Science Foundation's).

[5] Comité de Arquitectura de Internet, IAB (Internet Architecture Board).

todas ellas en EEUU. Paralelamente, otras redes troncales en Europa, tanto públicas como comerciales, junto con las americanas formaban el esqueleto básico ("backbone") de Internet.

A partir de 1989, con la integración de los protocolos OSI[6] en la arquitectura de Internet, se inició la tendencia actual de permitir no sólo la interconexión de redes de estructuras dispares, sino también facilitar el uso de distintos protocolos de comunicaciones.

En 1989 también, en el CERN[7] de Ginebra, Tim Berners-Lee y un grupo de físicos, crearon el lenguaje HTML, basado en el SGML[8]. En 1990 el mismo equipo construyó el primer cliente Web, llamado World Wide Web (WWW).

1.1.3 Direccionamiento IP y Servicios DNS

El direccionamiento IP pertenece a un protocolo de la capa de red, llamado *protocolo IP*, éste es el que se encarga del proceso de ruteo, es decir, del direccionamiento o encaminamiento de los paquetes a través de la red, hasta que lleguen a su destino. Todas las direcciones tienen un formato básico y no pueden ser tomadas de forma arbitraria.

Una dirección IP es la identificación de una máquina en concreto dentro de la red TCP/IP a la que pertenece y se divide en dos partes: el número de red y el número de host. El número de red es asignado por el Centro de información de Red Internet (InterNIC), mientras que el número de host es asignado por el administrador de la red. Un host es cualquier PC, servidor u otro componente que se pueda conectar a la red.

El tamaño de una dirección IP v4 es de 32 bits de longitud, tomada en 4 segmentos de 8 bits cada uno, se escribe de forma decimal y cada segmento se separa por puntos. Existen 5 clases diferentes de direcciones IP: clase A, clase B, clase C, clase D y clase E; Las clases A, B y C son de uso comercial, mientras que las clases D y E son para uso experimental o grupos de multidifusión.

---

[6] Basados en el Modelo OSI (Open Systems Interconnection), el cual esta formado por 7 capas que son: Física, Enlace de Datos, Red,Transporte, Sesión, Presentación y Aplicación.

[7] Centro Europeo de Investigación Nuclear o Laboratorio de Física de Partículas Elementales.

[8] El SGML (**Standard Generalized Markup Language**) es decir, Lenguaje de Señalización General Normalizado, es un sistema para la organización y etiquetado de documentos, en el cual está basado HTML.

Actualmente por la escasez de direcciones IP v4 se crearon direcciones IP v6, que tienen la misma función que las anteriores, pero están compuestas por 8 segmentos de 2 bytes cada uno, que suman un total de 128 bits, lo cual es equivalente a unos $3.4x10^{38}$ direcciones IP.

1.2 Introducción al Desarrollo en 3 Capas

La arquitectura de una aplicación es la vista conceptual de la estructura de ésta. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difieren según como está distribuido este código.

Los servicios son puestos en la red y operan de manera cooperativa para dar soporte a uno o más procesos de negocios. En este modelo, una aplicación se convierte en un conjunto de servicios de usuario, negocios y datos que satisfacen las necesidades de los procesos de negocios. Como los servicios están diseñados para el uso general y siguen lineamientos de interfaz publicados, pueden ser reutilizados y compartidos entre múltiples aplicaciones.

La arquitectura DNA[9] de tres capas cuenta con servicios específicos en cada capa, las cuales se comunican entre si mediante COM[10] (Component Object Model).

1.2.1 Capa de Presentación

Es la capa donde ésta la interfaz para el usuario, la parte que corresponde a la presentación, la cual le permite visualizar y acceder al sistema a través de una interfaz de amigable, la cual se compone típicamente de ventanas y un conjunto de controles, que pueden ser botones, cajas de texto, menús desplegables, cajas de dialogo, etc.

En ésta capa se desarrolla la presentación del sistema web, es decir la página principal del sistema y sus enlaces, son creados generalmente con lenguaje HTML y JavaScript, con el cual se realizan las comprobaciones y algunas operaciones del sistema.
Existen 3 tipos básicos de Interfaz que son:

[9] Arquitectura Microsoft Windows DNA la cual incorporan un amplio conjunto de servicios basados en la plataforma Windows.
[10] COM (Component Object Model) Arquitectura de Software que permite construir aplicaciones a partir de componentes de Software binarios, con el objeto de expandir las funciones del sistema operativo a nivel personalizado.

- **Interfaz clásica de ventanas**: Formada por una ventana principal y un menú desplegable.
- **Interfaz integrada en programas de gestión:** Esta consiste en una aplicación que se encuentra instalada en el equipo del usuario, un ejemplo es Microsoft Office, que contiene procesadores de texto y otras herramientas.
- **Navegadores:** Es una página web en el servidor que es descargada por el usuario en su máquina, ésta requiere un navegador como Internet Explorer o Netscape.

Los servicios de presentación proporcionan la interfaz necesaria para presentar información y reunir datos. Además aseguran los servicios de negocios necesarios para ofrecer las capacidades de transacciones requeridas e integrar al usuario con la aplicación.

La capa de servicios de presentación es responsable de:

- Obtener la información del usuario.
- Enviar la información del usuario a los servicios de negocios para su procesamiento.
- Recibir los resultados del procesamiento de los servicios de negocios.
- Presentar éstos resultados al usuario.

1.2.2 Capa de Negocios

En ésta capa se encuentran las reglas y la lógica de los procedimientos necesarios para realizar las operaciones del sistema.

Ésta capa interactúa con la Base de Datos y la capa de presentación, aquí se encuentran los cálculos, validaciones, controles, y demás funciones, del sistema, se encuentra del lado del servidor.

Es el "puente" entre un usuario y los servicios de datos, responde a peticiones del usuario (u otros servicios de negocios) para ejecutar una tarea de este tipo, cumple con esto aplicando procedimientos formales y reglas de negocio a los datos relevantes.

Una tarea de negocios es una operación definida por los requerimientos de la aplicación, como introducir una orden de compra o imprimir una lista de clientes. Las reglas de negocio son políticas que controlan el flujo de las tareas.

Cuando los datos necesarios residen en un servidor de bases de datos, garantizan los servicios de datos indispensables para cumplir con la tarea de negocios o aplicar su regla. Esto aísla al usuario de la interacción directa con la base de datos.

Como las reglas de negocio tienden a cambiar más frecuentemente que las tareas específicas de negocios a las que dan soporte, son candidatos ideales para encapsularlas en componentes que están lógicamente separados de la lógica de la aplicación en sí.

El nivel de servicios de negocios es responsable de:

- Recibir la entrada del nivel de presentación.
- Enviar el resultado procesado al nivel de presentación.
- Interactuar con los servicios de datos para ejecutar las operaciones de negocios para los que la aplicación fue diseñada a automatizar.

1.2.3 Capa de Datos

En esta capa se encuentran los mecanismos de acceso y control de los datos, que pueden tener vínculos a Base de Datos, a servidores de datos, etc.

El nivel de servicios de datos es responsable de:

- Almacenar los datos.
- Recuperar los datos.
- Mantener los datos.
- La integridad de los datos.

Los servicios de datos tienen una variedad de formas y tamaños, incluyendo los sistemas de administración de bases de datos relacionales (SABD), servidores de correo electrónico y sistemas de archivos tales como el NTFS[11].

1.3 Elementos de software para el desarrollo en WWW

1.3.1 Manejador de base de datos: Mysql

MySQL[12] es un sistema de administración de bases de datos relacionales, las cuales almacenan los datos en tablas separadas en lugar de poner todos los datos en un solo lugar.

[11] **NTFS (por siglas en inglés, New Technology File System) es un sistema de archivos diseñado por Microsoft, específicamente para Windows NT y Windows XP, con el objetivo de crear un sistema de archivos eficiente, robusto y con seguridad incorporada desde su base.**

Las tablas son enlazadas al definir relaciones que hacen posible combinar datos de varias tablas cuando se necesitan consultar datos. La ventaja de esto es que agrega velocidad y flexibilidad.

SQL[13] (Lenguaje Estructurado de Consulta) es el lenguaje más usado y estandarizado de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Reúne características del Álgebra y el Cálculo Relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una Base de Datos, de una forma sencilla.

1.3.2 Java Virtual Machine integrada al servidor: Jakarta Tomcat

Jakarta Tomcat fue creado y mantenido bajo soluciones código abierto en la plataforma Java. Esto implica que el servidor donde se instale tenga previamente instalada la plataforma Java (JSDK).

Tomcat funciona como un contenedor de servlets[14] desarrollado bajo el proyecto Jakarta[15] en la Apache Software Foundation[16]. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP)[17] de *Sun Microsystems*[18] y es considerado como un servidor de aplicaciones.

[12] www.mysql.com
[13] SQL es el lenguaje estándar para realizar el acceso a bases de datos.
[14]. Un *servlet* es un programa que se ejecuta en un servidor web, su uso más común es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web
[15] http://jakarta.apache.org/
[16] ASF (Apache Software Foundation) es una organización no lucrativa creada para dar soporte a los proyectos de software bajo la denominación ***Apache***, incluyendo el servidor web Apache.
[17] Los JSP contienen elementos HTML y código Java, que se ejecutan en un navegador web, son una abstracción de un servlet.
[18] http://java.sun.com

1.4 Propuesta del proyecto

Nombre del Proyecto: *Sistema de venta en línea para: CONSUMIBLES EN COMPUTACIÓN Y PAPELERÍA ÁNGELES.*

Resumen:

El proyecto se trata de un sistema para vender en Internet, los productos de papelería y consumibles de cómputo, está diseñado para realizar los pedidos a través del llenado de formularios, estos pedidos deben ser mayoristas.

El sistema sólo aceptará clientes mayoristas y empresas públicas o privadas.

Los usuarios del sistema deben identificarse con usuario y contraseña, en caso contrario sólo podrán acceder al catálogo más no podrán realizar pedidos, hasta no haberse identificado en el sistema.

También contará con una parte de mantenimiento del sistema, para dar de alta los productos y actualizar catálogos y precios, pero ésta sólo será accesible y visible para el administrador del sistema.

Tiempos asociados al proyecto: Inicio: 08/Septiembre/2004.
Fin: 20/Mayo/2005.

1.4.1 Descripción del proyecto

Descripción de la organización:

Empresa dedicada a la venta de artículos de oficina, equipos y consumibles de cómputo y papelería en general. Surte pedidos a empresas privadas, así como a secretarías de gobierno. Ofrece precios de mayoreo y puede ajustarse a los precios de la competencia.

Su nombre es *"Consumibles de Computación y Papelería Ángeles"*. Cuenta con una oficina, en la cual se hacen los pedidos y una papelería abierta al público en general.

- El pedido se realiza vía telefónica o por fax y se surte 1 día después.
- No existen intermediarios en la entrega de pedidos, se hacen personalmente.
- La forma de pago puede ser de contado o a crédito, de 15 a 30 días a partir de la fecha en que el cliente meta a revisión sus facturas.

La papelería cuenta con artículos propios de este tipo de negocios, ofrece además productos como juguetes (juegos de mesa, pelotas, etc.), regalos y artículos de mercería (listones, etc.). Brinda servicios de enmicado, engargolado y fotocopiado.

1.4.2 Requerimientos del sistema:

Se requiere diseñar la infraestructura de un sistema que permita hacer más eficiente el servicio de solicitud y realización de pedidos de la empresa, mediante la adecuada administración y control de existencias de los artículos, así como de los diferentes tipos de clientes, para obtener el máximo aprovechamiento de tiempo y servicio.

"Consumibles de Computación y Papelería Ángeles"

¿Qué hace?

- Vende artículos de oficina, equipos y consumibles de cómputo y papelería en general a empresas privadas o secretarías de gobierno que necesitan este tipo de productos.
- Cuenta con una papelería abierta al público en general, en la cual se venden artículos propios de este tipo de negocios, cuenta también con regalos, juguetes y algunos artículos de mercería. Ofrece además servicios de enmicado, engargolado y fotocopiado.

¿Cómo?

Vía Telefónica: (El cliente llama).

- Se identifica al cliente (público en general, diferentes tipos de empresas, clientes frecuentes).
- Solicita el pedido.
- Se confirma el pedido.
- Se verifica la existencia de los artículos (en caso de que falte alguno, se procede a conseguirlo inmediatamente).
- Se completa el pedido.
- Se envía el pedido al siguiente día junto con la factura, la cual describe las características del pedido (artículos, cantidad, precios, subtotal, IVA y total a pagar).
- Se entrega el pedido y la factura (los cuales serán revisados por el cliente).
- Se pregunta la forma de pago.
- Si la forma de pago es de contado, se realiza el pago inmediatamente.

- Si la forma de pago es a crédito, se le da un plazo a pagar de 15 a 30 días (a partir de la fecha que el cliente meta a revisión sus facturas).
- Si la forma de pago es en efectivo, se realiza el pago en ese momento.
- Si la forma de pago es por cheque, el cliente lo entrega y se procede a cobrarlo.
- Si la forma de pago es por transacción bancaria, se revisa el estado bancario de la cuenta de la empresa para saber si el cliente ya realizó la transacción.

Vía fax: (La empresa manda cotización por fax).

- La empresa manda a sus clientes cotización de los artículos por vía fax.
- El cliente regresa el fax con una selección de artículos, los cuales conforman su pedido.
- Se verifica la existencia de los artículos, en caso de que falte alguno, se procede a conseguirlo inmediatamente.
- Se completa el pedido.
- Se envía el pedido al siguiente día junto con la factura, la cual describe las características del pedido (artículos, cantidad, precios, subtotal, IVA y total a pagar).
- Se entrega el pedido y la factura (los cuales serán revisados por el cliente).
- Se pregunta la forma de pago.
- Si la forma de pago es de contado, se realiza el pago inmediatamente.
- Si la forma de pago es a crédito, se le da un plazo a pagar de 15 a 30 días, a partir de la fecha que el cliente meta a revisión sus facturas.
- Si la forma de pago es en efectivo, se realiza el pago en ese momento.
- Si la forma de pago es por cheque, el cliente lo entrega y se procede a cobrarlo.
- Si la forma de pago es por transacción bancaria, se revisa el estado bancario de la cuenta de la empresa para saber si el cliente ya realizó la transacción.

Eficientar:

- Manejar un mayor número de artículos.

- Obtener más clientes.

Conceptualizar:

- Nuevos servicios (catálogo de productos, categorías y precios).
- Precios especiales a clientes frecuentes.
- Entrega de mercancía en el menor tiempo.
- Garantizar precios bajos.

Back Office:

Solicitud del registro del cliente:

- Nombre de la empresa.
- Tipo de la empresa (privada o de gobierno).
- RFC.
- Domicilio.
- Teléfono.
- Contacto.
- Tipo de artículos que le interesan.
- Cantidad aproximada del pedido.
- Forma de pago.

Realizar pedido:

- Identificar al cliente (con sus datos generales como nombre, RFC, etc.).
- Datos del pedido (como fecha de solicitud, artículos, cantidad, precios, subtotal, IVA, total a pagar, fecha de entrega).
- Verificación de artículos disponibles.
- Completar el pedido.
- Elaboración de factura.

Consulta de los artículos disponibles:

- Comparar las salidas y entradas de los artículos en el almacén.
- Si no hay existencias del producto, conseguirlo de inmediato con algún proveedor.

Cobro del pedido en efectivo:

- Elaboración de la factura.
- Entrega de la factura al cliente.

- Revisión de la factura por el cliente.
- Si la forma de pago es de contado, se realiza el pago inmediatamente.
- Si la forma de pago es a crédito, se le da un plazo a pagar de 15 a 30 días, a partir de la fecha que el cliente meta a revisión sus facturas. Después de ésta fecha se cobrará personalmente la factura
- Entrega de la factura pagada.

Cobro del pedido por cheque:

- Elaboración de la factura.
- Entrega de la factura al cliente.
- Revisión de la factura por el cliente.
- Si la forma de pago es de contado, se entrega el cheque con la cantidad completa en ese momento y se procede a cobrar el cheque en el banco.
- Si la forma de pago es a crédito, se le da un plazo a pagar de 15 a 30 días, a partir de la fecha que el cliente meta a revisión sus facturas. Después de ésta fecha se procederá a cobrar el cheque en el banco.
- Entrega de la factura pagada.

Cobro del pedido por transferencia bancaria:

- Elaboración de la factura.
- Entrega de la factura al cliente.
- Revisión de la factura por el cliente.
- Si la forma de pago es de contado, se revisará el estado bancario de la cuenta de la empresa para saber si el cliente ya realizó la transacción completa.
- Si la forma de pago es a crédito, se le da un plazo a pagar de 15 a 30 días, a partir de la fecha que el cliente meta a revisión sus facturas. Después de ésta fecha se revisará el estado bancario de la cuenta de la empresa para saber si el cliente ya realizó la transacción completa.
- Entrega de la factura pagada.

1.4.3 Objetivo del proyecto

¿Cuál es el objetivo del proyecto?

Hacer una aplicación para realizar pedidos y vender en línea, los diversos artículos de papelería y productos consumibles de cómputo ofrecidos por *"CONSUMIBLES DE COMPUTACIÓN Y PAPELERÍA ÁNGELES"*.

¿Cuáles son las características particulares y los beneficios del producto?

- Un sistema de realización de pedidos reutilizable para diversos tipos de papelerías. Permitirá categorizar productos, precios, descuentos, control de seguridad en los pedidos.
- El sistema de realización de pedidos contendrá muchas secciones configurables, lo que lo hacen de fácil crecimiento y adaptación.
- El sistema de realización de pedidos y venta en línea dará acceso exclusivo a usuarios registrados, una vez que se inicien las operaciones. Mientras sean solo consultas dará acceso a público en general.
- El sistema utilizará un esquema de seguridad para garantizar diversos tipos de acceso, de acuerdo con el tipo de usuario identificado.

1.4.4 Alcance

Queremos enfocarnos en el diseño de una aplicación web con capacidad para manejar los pedidos en línea y realizar transacciones en la Base de Datos de forma confiable. Para contar con un servicio configurable, amigable, extensible y que se mantenga vigente con el tiempo y con el crecimiento.

En alcance	Fuera del alcance
Construir una aplicación que pueda usarse con servidores estándares de Web y de aplicación.	Construir un nuevo servidor de aplicaciones Web.
Hacer que la interfaz sea compatible con los navegadores Internet Explorer y Mozilla.	Mantener compatibilidad con navegadores poco comunes o de versiones muy atrasadas.
Seguridad en el sistema a través de usuarios/contraseñas y niveles de acceso	Protección total de la aplicación contra ataques de *"Hackers"*.[19]

[19] Programadores maliciosos que realizan ataques utilizando las vulnerabilidades de los sistemas, introduciendo virus, robando información de cuentas, etc.

En alcance	Fuera del alcance
Una interfaz de usuario sencilla y amigable.	Una lista sofisticada de "skins" para adaptar la interfaz de usuario a cada usuario en particular.
Niveles de servicio en base de datos que puedan trabajarse en un solo servidor.	Acceso a un "cluster" de bases de datos y servidores de transacciones para garantizar 24x7x365.
Desplegar *"banners"*[20] publicitarios a los visitantes en zonas definidas para ello.	Enviar publicidad dirigida de acuerdo al tipo y perfil de usuario que se encuentra dentro de la aplicación.

1.4.5 Entregables

Listar los productos entregables del proyecto al momento de concluir.

- Aplicación web para "Venta de productos de papelería y consumibles en línea"
- Guía de instalación y configuración
- Muestra de la interfaz de usuario
- Ayuda en línea para usuarios de la aplicación
- Configuración de la zona de *"banners"* publicitarios y generación de reportes del sitio web

1.4.6 Riesgos y Beneficios

Riesgos

1. Existe el riesgo de que el sistema de pedidos en línea, no sea fácil de configurar, por personal no capacitado, para evitar esto se incluirá el manual técnico del sistema.
2. Riesgo de acceso ilegal a la base de datos.

Beneficios

Si cumplimos el alcance del sistema nuestra aplicación dará un beneficio en ahorro operativo y mejora en calidad de servicio a los clientes que realizan pedidos y adquieren productos a gran escala a través un sitio web para venta de productos consumibles y papelería en general.

[20] Elementos emergentes en las páginas web que sirven como medios publicitarios.

CAPÍTULO 2. CONCEPTOS PARA EL ANÁLISIS Y DISEÑO DE UN PROYECTO ORIENTADO A WEB

2.1 Enfoque orientado a objetos

El enfoque orientado a objetos, es una forma de observar la realidad, que se basa en la definición de objeto:

> *Un objeto es todo aquello que tiene características propias que lo hacen único e indivisible dentro del entorno al que pertenece. También se puede definir como una entidad compleja provista de datos (propiedades, atributos) y comportamiento (funcionalidad, programas, métodos) que corresponden a los objetos reales del mundo que nos rodea.*

2.1.1 Estructura de un objeto

Un objeto posee:

a) *Relaciones*: Son las que permiten que el objeto se inserte en un grupo y están formadas por vínculos o punteros a otros objetos.

b) *Propiedades*: Son aquellos atributos o características que distinguen al objeto de los demás objetos de un mismo grupo, estas propiedades pueden ser heredadas.

c) *Métodos*: Son las operaciones que pueden realizarse sobre el objeto, éstos son fragmentos de código o programas que el objeto es capaz de ejecutar y pueden acceder a ellos sus descendientes por herencia.

2.1.2 Características de los lenguajes orientados a objetos

2.1.2.1 Abstracción

Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar *cómo* se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando los están, una variedad de técnicas son requeridas para ampliar una abstracción.

2.1.2.2 Encapsulación

También llamada "ocultación de la información", esto asegura que los objetos no pueden cambiar el estado interno de otros objetos de maneras inesperadas;

solamente los propios métodos internos del objeto pueden acceder a su estado.

2.1.2.3 Polimorfismo

Las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del referente.

2.1.2.4 Herencia

Organiza y facilita el polimorfismo y la encapsulación permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir y extender su comportamiento sin tener que volver a implementar su comportamiento. Esto suele hacerse habitualmente agrupando los objetos en *clases* y las clases en *árboles* o *enrejados* que reflejan un comportamiento común.

2.2 Ciclo de vida de los sistemas de información

2.2.1 Encuesta o Entrevista

Es una plática formal donde se preguntan las necesidades del cliente y sus requerimientos. Es un cuestionario donde se hacen las preguntas claves: ¿Qué necesitas?, ¿Qué quieres que haga? y ¿Cómo quieres que lo haga?

Es un documento escrito y al final se debe firmar por ambos, el cliente y el desarrollador, para quedar de acuerdo y evitar conflictos a momento de la entrega del proyecto.

2.2.2 Análisis

Consiste en analizar el entorno de propio sistema, el hardware y el software requerido y el existente, los costos, es decir, ¿qué?, ¿con qué? Y ¿porqué? Y hacer un análisis de factibilidad, hasta llegar al sí hacerlo. Para obtener los siguientes productos documentales:

- Encuesta: es un documento escrito donde se presentan las necesidades y los requerimientos del cliente.
- Especificación de procesos: Es la lista de los procesos o tareas fundamentales que el sistema realiza; ¿qué realiza?, ¿cómo lo realiza? y ¿quién lo realiza?
- Diagrama entidad relación: Es el diagrama de la Base de Datos del sistema,

donde se modela cada taba y cada entidad de la Base de Datos y las relaciones entre ellas.

- Diccionario de datos: Es un documento donde se describe cada tabla y cada campo, lo que contienen, sus tipos de datos que almacenan, etc.
- Casos de uso del sistema: Es la forma gráfica de la especificación de procesos, donde se establecen los actores en cada caso y sus funciones..
- Manual técnico y de usuario: El manual de usuario sirve para que los usuarios aprendan a utilizar el sistema y el técnico, sirve para que los desarrolladores sepan que hacer si ocurre algún problema en el sistema y puedan corregirlo.

2.3 Diseño de la Base de Datos

2.3.1 Recomendaciones para el diseño de la Base de Datos

a. Identificar el mayor número de entidades posibles que estén involucradas en el sistema de información.
b. Encontrar el mayor número de atributos para cada entidad.
c. Buscar siempre relacionar todas las entidades del sistema. Si una entidad no tienen relación con al menos una tabla no debe existir en la BD.
d. Buscar siempre que los atributos de una tabla sean numéricos.
e. El nombre de las entidades debe ser singular, significativo y en minúsculas.
f. El nombre de cada atributo debe tener relación con la tabla a la que pertenece, para ello añadir un prefijo que haga referencia a la tabla.
g. Cuando una entidad tiene más de un valor posible para uno de sus atributos, éste atributo se convierte en una entidad.
h. Manejar llaves primarias, una llave primaria es un conjunto de campos que permiten garantizar que el registro sea único.

2.4 Proceso de Análisis de Requerimientos

2.4.1 Modelo de casos

Éste modelo extrae el conocimiento funcional del problema de una forma estructurada y progresiva, siendo la base para establecer la estructura del sistema. En este modelo se establecen las principales transacciones que contendrá el sistema, es

decir que cada interacción que el sistema tendrá con los agentes externos a él que se denominarán Actores. Cada transacción del sistema recibe el nombre de Caso y cada caso requiere una especificación, tanto de nombre como de secuencia de pasos necesarios para llevarlo a cabo.

2.4.1.1 Casos de uso

Es el que describe el comportamiento del sistema bajo condiciones variantes en las que interactúan con los actores. Un actor es cualquier persona o cosa que interactúa con el sistema. Es fundamentalmente texto y sirve para comunicarse entre personas, que no tienen conocimientos técnicos, ésta escrito en lenguaje natural y ésta libre de tecnología. Pueden ser tan generales o específicos como se desee y dirigen la forma de hacer que los requerimientos funcionales sean legibles.

Un caso de uso presenta un escenario, en él un actor tiene objetivos por cumplir, los que nombran los casos de uso, es decir, el nombre del caso de uso es la sentencia del objetivo por cumplir. El caso de uso reúne objetivos y escenarios en un solo elemento.

2.4.1.2 Cómo elaborar casos de uso

1) Identificar actores y metas. Un actor es cualquier cosa con un comportamiento.
2) Para cada caso de uso:
   a) Escribir el objetivo del caso de forma simple
   b) Escribir el escenario principal de éxito
   c) Capturar cada intento del actor por invocar funciones de sistema para resolver requerimientos
   d) Identificar que información intercambian
3) Escribir condiciones de fallo como extensiones o excepciones. Usualmente cada caso puede fallar.
4) Anotar la condición de fallo de forma separada, después de la condición de éxito. Del lado izquierdo el éxito, del lado derecho fallo o flujo alterno.
5) Para cada condición de fallo.
   a) Seguir la falla hasta que se concluya o se recupere el fallo. Algunas condiciones de fallo permiten recuperar el camino al flujo principal de éxito.
   b) Todos los escenario deben tener un inicio y fin definidos.

6) El valor de escenario de fallo es detectar situaciones inusuales o incompletas.

2.4.2 Modelo de interfaz

Establece el vínculo visual entre el desarrollador y el usuario para concretar aspectos de la interacción que el sistema pudiese tener con su entorno. Se compone de la definición de interfaces principales, es decir, Pantallas, reportes y llamadas a otros sistemas, que participarán en la ejecución de cada caso de uso. Este modelo se realiza en forma simultánea al modelo de casos ya que están relacionados.

2.4.3 Modelo de dominio del problema

En este se establecerán los principales objetos que constituirán al sistema y las relaciones entre sí. Tiene como objetivo identificar los objetos de información y las relaciones que tienen entre sí. En este modelo se realiza un Diagrama de Clases por cada caso del Modelo de Casos.

2.5 UML

UML (Lenguaje Unificado de Modelado) es un lenguaje gráfico para visualizar, especificar, construir y documentar los componentes de un sistema de software, cuenta con varios tipos de modelos, los cuales muestran diferentes aspectos de las entidades representadas, permite tanto la especificación conceptual de un sistema como la especificación de elementos concretos, como pueden ser las clases o un diseño de base de datos.

Según su definición, los objetivos de UML son los siguientes:

- Visualizar: UML permite representar mediante su simbología el contenido y la estructura de un sistema software. La notación UML permite definir modelos que serán claramente comprensibles por otros desarrolladores facilitando así el mantenimiento del sistema que describe.
- Especificar: UML permite especificar los procesos de análisis, diseño y codificación de un sistema software. También permite determinar modelos precisos, sin ambigüedades, detallando las partes esenciales de los mismos.
- Construir: Las anteriores características permiten que UML pueda generar código en distintos lenguajes de programación y tablas en una base de datos a partir de modelos UML. Además permite simular el comportamiento de sistemas software.

- Documentar: Como ya se comentó antes, UML permite especificar los procesos de análisis, diseño y codificación y también permite documentar los mismos, dejando clara la arquitectura del sistema.

2.5.1 Diagrama de Casos de uso

Es la representación gráfica del caso de uso, donde los actores se representan con una figura de alambre, los casos se representan con un ovalo y las relaciones son líneas.

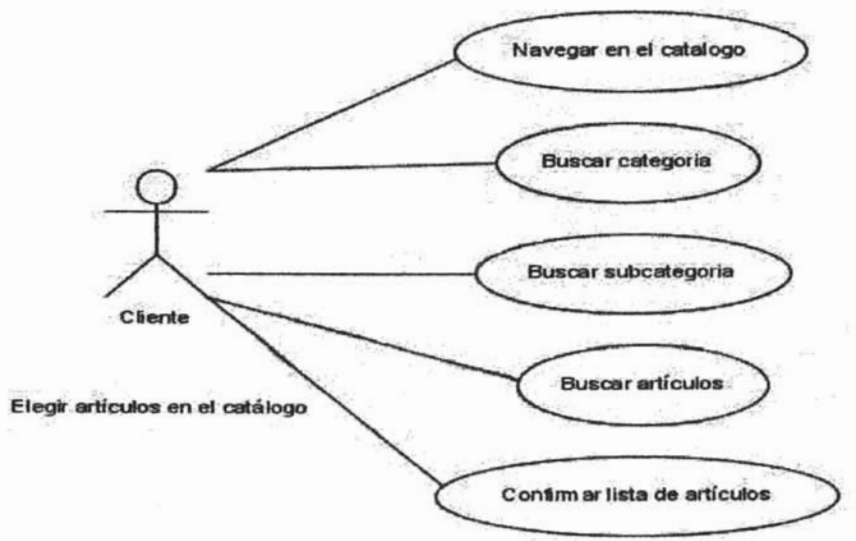


Fig. 1 Caso de uso: Elegir artículo de Catálogo.

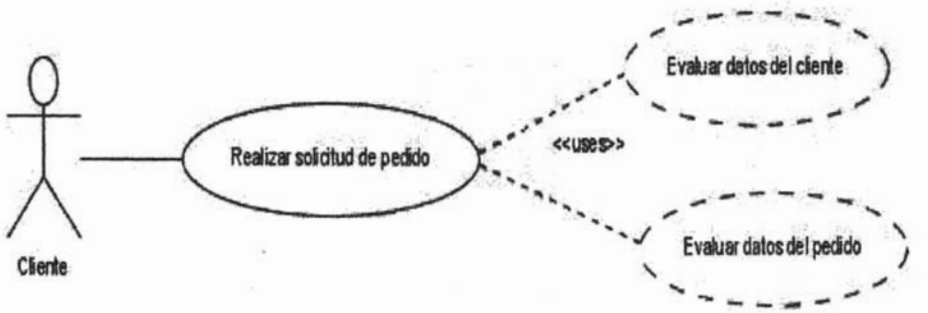


Fig. 2 Caso de uso: Solicitar pedido.

2.5.2 Diagrama de Secuencia

Modela la interacción ordenada entre los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia de tiempo. El eje vertical representa el tiempo y en el horizontal se colocan los objetos y actores participantes en la interacción. Cada objeto o actor tiene una línea vertical. Los mensajes se representan mediante flechas entre los distintos objetos.

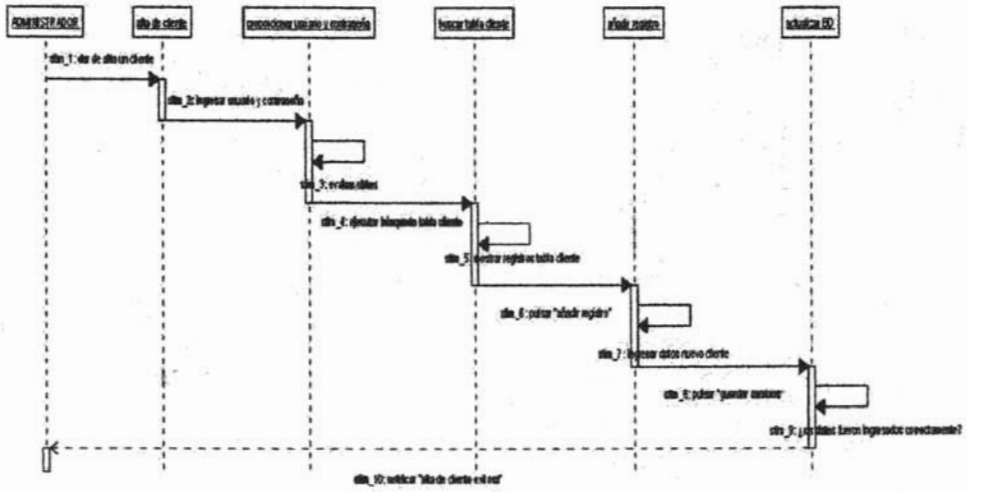


Fig. 3 Diagrama Secuencia Alta de Cliente

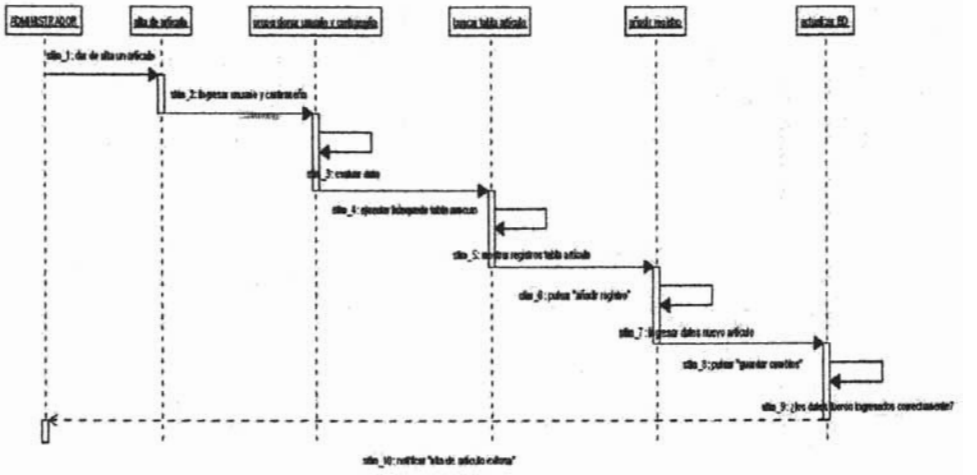


Fig. 4 Diagrama de Secuencia Alta producto

2.5.3 Diagrama de Clases

Muestra un conjunto de clases y sus colaboraciones y relaciones. Estos diagramas sirven para visualizar las relaciones existentes entre las distintas clases y la forma en que colaboran unas con otras.

2.5.4 Diagrama de Estado

Es la secuencia de estados por los que pasa un caso de uso, o un objeto a lo largo de su vida o todo el sistema, indica que eventos hacen que pase de un estado a otro y cuales son las respuesta y acciones que genera. La representación de un diagrama de estados es un gráfo cuyos nodos son estados y los arcos son transacciones; un estado se representa como una caja redondeada con el nombre del estado en su interior y una transición se representa como una flecha desde el estado origen al estado destino.

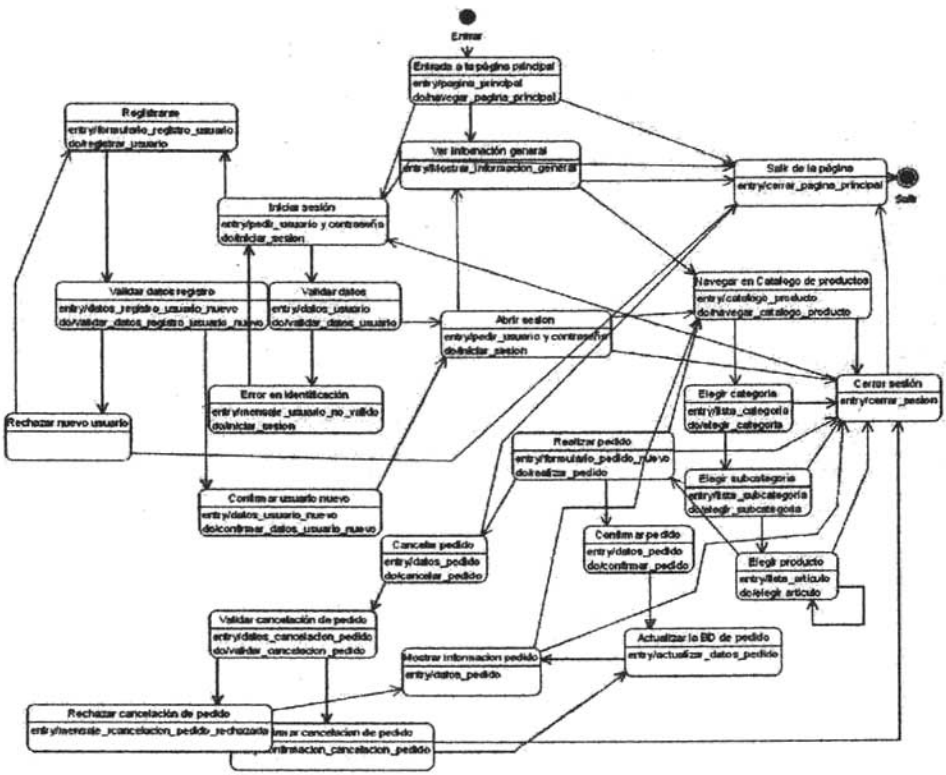


Fig. 4 Diagrama de Estado del Sistema

2.6 Análisis del Proyecto

Introducción

La aplicación es un sistema para vender en Internet, los productos de papelería y consumibles de computo, esta diseñado para realizar los pedidos por Internet, a través del llenado de formularios, estos pedidos deben ser mayoristas. Los usuarios del sistema se deberán identificar con un nombre y contraseña para poder acceder a las funciones especificas del sistema, en caso de no estar registrado en el sistema se podrá registrar como nuevo usuario, su solicitud será enviada y por correo se confirmará su registro proporcionándole un nombre y contraseña. El sistema solo aceptara clientes mayoristas y a empresas privadas, públicas o gubernamentales.

También contará con una parte de mantenimiento del sistema, para dar de alta los productos y actualizar catálogos y precios, pero esta parte solo será accesible y visible para el administrador del sistema, quién deberá identificarse con un usuario y contraseña administrador.

2.6.1 Requerimientos funcionales

Requerimiento del sistema:

Se requiere diseñar la infraestructura de un sistema que permita hacer más eficiente el servicio de solicitud y realización de pedidos de la empresa, mediante la adecuada administración y control de existencias de los artículos, así como de los diferentes tipos de clientes, para obtener el máximo aprovechamiento de tiempo y servicio.

Actores

1. Clientes (empresas privadas o gubernamentales)
2. Administrador de la papelería.

Roles

Clientes (empresas privadas o gubernamentales y clientes frecuentes):

- Navega por la página web para armar su "carrito".
- Solicita el pedido.
- Selecciona el tipo de pago.
- Realiza el pago.

Administrador de la empresa:

- Define y da de alta o de baja las categorías de los productos.

- Define y da de alta o de baja los productos.
- Define y da de alta o de baja los clientes.
- Define y da de alta o de baja los proveedores.
- Define la cantidad mínima de dinero para poder solicitar un pedido.
- Selecciona el tipo de clientes (para dar precios especiales).

Sistema (proyecto):

- Muestra los artículos con sus diferentes categorías y precios.
- Registra a los clientes diferenciándolos (empresas privadas o gubernamentales, pequeñas o grandes y clientes frecuentes).
- Compara las entradas y salidas de los artículos.
- Compara las altas y bajas de los clientes, proveedores y artículos.

Sistema especializado:

- Confirma los pedidos.
- Realiza pago.

2.6.2 Documentación de Casos de Uso

Caso de uso #01.

Nombre: "Añadir un cliente".
Precondiciones: El cliente (combinación de sus datos) no debe existir en la base de datos.
Poscondiciones: El administrador habrá añadido un nuevo cliente a la base de datos.
Inicio: Cuando el administrador desea añadir un cliente en la base de datos.
Fin: Al añadir el nuevo cliente en la base de datos.
Flujo normal:.
1. El administrador se identifica en el sistema
2. El administrador accede a la tabla de clientes.
3. El administrador escribe cada uno de los datos del cliente en la base de datos (razón social, RFC, domicilio fiscal, teléfono, etc)
4. El administrador pulsa "guardar cambios".
5. El sistema guarda los datos introducidos, actualiza los cambios en la base de datos. El cliente ha sido añadido.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron

algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. Los datos fueron ingresados incorrectamente. El sistema le envía un mensaje de error comunicándoselo e indicándole cuáles campos fueron mal ingresados.
4. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
5. El proveedor ya existe. Cancelar acción de añadir cliente.

Caso de uso #02.

Nombre:"Actualizar información de un cliente".
Precondiciones:. El cliente debe existir en la base de datos
Poscondiciones: El administrador habrá actualizado los datos de un cliente en la base de datos.
Inicio: Cuando el administrador desea actualizar la información de un cliente en la base de datos.
Fin: Al actualizar la información de un cliente en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador hace una modificar en el sistema.
3. El sistema le muestra los datos completos del cliente.
4. El administrador actualiza los campos búsqueda del cliente que quiere que le interesan.
5. El administrador pulsa "guardar cambios".
6. El sistema actualiza los cambios en la base de datos. Uno o varios datos del cliente han sido actualizados en la base de datos.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de"Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.

3. No existe ningún cliente que coincida con la búsqueda. Se cancela la actualización o bien se procede a añadir un nuevo cliente con todas sus datos.
4. Los datos fueron ingresados incorrectamente. El sistema le envía un mensaje de error comunicándoselo e indicándole cuáles campos fueron mal ingresados.
5. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
6. El cliente ya está actualizado. Cancelar acción de actualizar cliente.

Caso de uso #03.

Nombre: "Dar de baja un cliente".
Precondiciones: El cliente (combinación de sus datos) debe existir en la base de datos.
Poscondiciones: El administrador habrá dado de baja a un cliente en la base de datos.
Inicio: Cuando el administrador desea dar de baja a un cliente de la base de datos.
Fin: Al dar de baja un cliente en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador hace una búsqueda del cliente que quiere dar de baja en el sistema.
3. El sistema le muestra los datos completos del cliente.
4. El administrador selecciona el campo de "status" y escribe en este "0" (baja).
5. El sistema le preguntará si desea guardar los cambios.
6. El administrador pulsa "aceptar".
7. El sistema actualiza la base de datos. El cliente ha sido dado de baja en la base de datos.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
4. No existe ningún cliente que coincida con la búsqueda. Cancelar acción de dar

de baja a un cliente.

Caso de Uso #04.

Nombre: "Añadir una categoría".
Precondiciones: La categoría no debe existir en la base de datos.
Poscondiciones: El administrador habrá añadido una categoría en la base de datos.
Inicio: Cuando el administrador desea añadir una categoría en la base de datos.
Fin: Al añadir una nueva categoría en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador accede a la tabla de categorías.
3. El administrador escribe el nombre de la nueva categoría (con todas sus características) en la base de datos.
4. El administrador pulsa "guardar cambios".
5. El sistema guarda los datos introducidos, actualiza los cambios en la base de datos. La categoría ha sido añadida.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. Los datos fueron ingresados incorrectamente. El sistema le envía un mensaje de error comunicándoselo e indicándole cuáles campos fueron mal ingresados.
4. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
5. La categoría ya existe. Cancelar acción de añadir categoría.

Caso de uso #05.

Nombre: "Actualizar información de una categoría".
Precondiciones: La categoría debe existir en la base de datos.
Poscondiciones: El administrador habrá actualizado una categoría en la base de datos.
Inicio: Cuando el administrador desea actualizar la información de una categoría en la base de datos.

Fin: Al actualizar la información de una categoría en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador hace una búsqueda de la categoría que desea modificar en el sistema. 3. El sistema le muestra los datos completos de la categoría.
4. El administrador actualiza los campos que le interesan.
5. El administrador pulsa "guardar cambios".
6. El sistema actualiza los cambios en la base de datos. Uno o varios datos de la categoría han sido actualizados en la base de datos.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo. 2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. Los datos fueron ingresados incorrectamente. El sistema le envía un mensaje de error comunicándoselo e indicándole cuáles campos fueron mal ingresados.
4. No existe ninguna categoría que coincida con la búsqueda. Se cancela la actualización o bien se procede a añadir un nueva categoría con todas sus características.
5. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
6. La categoría ya está actualizada. Cancelar acción de actualizar categoría.

Caso de uso #06.

Nombre: "Dar de baja una categoría".
Precondiciones: La categoría debe existir en la base de datos.
Poscondiciones: El administrador habrá dado de baja una categoría en la base de datos.
Inicio: Cuando el administrador desea dar de baja una categoría en la base de datos.
Fin: Al dar de baja una categoría en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador hace una búsqueda de la categoría que quiere dar de baja en

el sistema.
3. El sistema le muestra los datos completos de la categoría.
4. El administrador selecciona el campo de "status" y escribe en este "0" (baja).
5. El sistema le preguntará si desea guardar los cambios.
6. El administrador pulsa "aceptar".
7. El sistema actualiza la base de datos. La categoría ha sido dado de baja en la base de datos.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
4. No existe ninguna categoría que coincida con la búsqueda. Cancelar acción de dar de baja la categoría

Caso de uso #07.

Nombre: "Añadir un artículo".
Precondiciones: El artículo (combinación de sus características) no debe existir en la base de datos.
Poscondiciones: El administrador habrá añadido un artículo a la base de datos.
Inicio: Cuando el administrador desea añadir un artículo en la base de datos.
Fin: Al añadir el nuevo artículo en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador accede a la categoría donde se quiere añadir el producto.
3. El administrador accede a la subcategoría donde se quiere añadir el producto.
4. El administrador escribe cada una de sus características en la base de datos (nombre, descripción, marca, precio unitario, etc)
5. El administrador pulsa "guardar cambios".
6. El sistema guarda los datos introducidos, actualiza los cambios en la base de datos. El artículo ha sido añadido al catálogo.
Flujos de excepción:

1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. No existe una categoría afín para el producto. El administrador debe crear una nueva categoría (si es necesario) en el sistema.
4. Los datos fueron ingresados incorrectamente. El sistema le envía un mensaje de error comunicándoselo e indicándole cuáles campos fueron mal ingresados.
5. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
6. El producto ya existe. Cancelar acción de añadir producto.

Caso de uso #08.

Nombre: "Actualizar información de un artículo".
Precondiciones: El artículo debe existir en la base de datos.
Poscondiciones: El administrador habrá actualizado un artículo en la base de datos.
Inicio: Cuando el administrador desea actualizar la información de un artículo en la base de datos.
Fin: Al actualizar la información de un artículo en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador hace una búsqueda del artículo que quiere modificar en el sistema.
3. El sistema le muestra los datos completos del artículo.
4. El administrador actualiza los campos que le interesan.
5. El administrador pulsa "guardar cambios".
6. El sistema actualiza los cambios en la base de datos. Uno o varios datos del artículo han sido actualizados en el catálogo.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un

determinado tiempo.
3. Los datos fueron ingresados incorrectamente. El sistema le envía un mensaje de error comunicándoselo e indicándole cuáles campos fueron mal ingresados.
4. No existe ningún artículo que coincida con la búsqueda. Se cancela la actualización o bien se procede a añadir un nuevo producto con todas sus características.
5. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.
6. El producto ya está actualizado. Cancelar acción de actualizar producto.

Caso de uso #09.

Nombre: "Dar de baja un artículo".
Precondiciones: El artículo (combinación de sus características) debe existir en la base de datos.
Poscondiciones: El administrador habrá dado de baja a un artículo en la base de datos.
Inicio: Cuando el administrador desea dar de baja un artículo en la base de datos.
Fin: Al dar de baja un artículo en la base de datos.
Flujo normal:
1. El administrador se identifica en el sistema.
2. El administrador hace una búsqueda del artículo que quiere dar de baja en el sistema.
3. El sistema le muestra los datos completos del artículo.
4. El administrador selecciona el campo de "status" y escribe en este "0" (baja).
5. El sistema le preguntará si desea guardar los cambios.
6. El administrador pulsa "aceptar".
7. El sistema actualiza la base de datos. El artículo ha sido dado de baja en la base de datos.
Flujos de excepción:
1. El administrador ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el administrador ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. Hubo un error al guardar los cambios. Se realiza de nuevo el proceso.

4. No existe ningún artículo que coincida con la búsqueda. Cancelar acción de dar de baja el artículo.

Caso de uso #10.

Nombre: "Solicitar registro de cliente".
Precondiciones: El usuario no debe ser un cliente registrado.
Poscondiciones: El usuario será un cliente registrado.
Inicio: Al solicitar el registro de un nuevo usuario como cliente.
Fin: Al recibir el nombre de usuario y contraseña por parte del sistema.
Flujo normal:
1. El usuario pulsa en la opción "registro de cliente".
2. El sistema muestra el formulario de registro.
3. El usuario llena el formulario de registro.
4. El usuario pulsa la opción de "enviar formulario".
5. El sistema envía formulario de registro.
6. El sistema muestra la confirmación del envío del formulario.
7. El sistema confirma los datos del formulario de registro.
8. El sistema registra al nuevo usuario en la base de datos.
9. El sistema envía nombre de usuario y contraseña.
10. El usuario recibe el nombre de usuario y contraseña.
Flujos de excepción:
1. El usuario llenó algunos campos incorrectamente. Se manda un mensaje de error comunicándoselo e indicándole cuales son los campos en los que se ha equivocado, para que pueda corregirlos.
2. El usuario no llenó los campos obligatorios. Se manda un mensaje de error comunicándoselo e indicándole cuales son los campos que son necesarios para el registro.
3. El envío del formulario no ha sido satisfactorio. Se manda un mensaje de error comunicándoselo y se le invita a volver a mandarlo.
4. El usuario no llena los campos. Cerrar cesión después de un determinado tiempo.
5. El usuario pulsa en la opción "cancelar". El sistema lo envía a la página principal.
6. El usuario ya es un usuario registrado, se manda un mensaje de error comunicándoselo.

7. El usuario no pulsa "enviar formulario". Cerrar cesión después de un determinado tiempo.

Caso de uso #11.

Nombre: "Identificar al cliente".
Precondiciones: El usuario habrá ingresado a la página web de la empresa.
Poscondiciones: El usuario será identificado correctamente por el sistema.
Inicio: Al ingresar nombre de usuario y contraseña.
Fin: Al identificar al cliente con su nombre de usuario y contraseña correctos.
Flujo normal:
1. El usuario ingresa al portal web de la empresa.
2. El sistema muestra los campos para identificarse como cliente.
3. El usuario llena los campos con su nombre de usuario y contraseña.
4. El usuario pulsa la opción de "aceptar".
5. El sistema compara que el nombre de usuario y contraseña sean correctos.
6. El usuario es identificado por el sistema.
Flujos de excepción:
1. El usuario ingresó los datos incorrectamente (le faltaron o sobraron algún(os) caracteres o tiene la tecla de "Caps Lock" encendida). El sistema le envía un error comunicándoselo y pidiéndole se identifique de nuevo.
2. Es la tercera vez que el usuario ingresa un nombre no válido o contraseña incorrecta. El sistema bloqueará ese nombre de usuario por un determinado tiempo.
3. El usuario no recuerda su contraseña. El usuario pulsa la opción "no recuerdo mi contraseña", el sistema lo enviará a una página en donde habrá algunos campos que deberá llenar (información que sólo el cliente pudiera contestar), pulsará después "aceptar" y el sistema le enviará la contraseña a su correo electrónico.

Caso de uso #12.

Nombre: "Elegir artículos en el catálogo".
Precondiciones: El usuario deberá ser un cliente registrado.
Poscondiciones: El usuario tendrá una lista de artículos elegidos.
Inicio: Cuando el cliente desea elegir uno o varios artículos.
Fin: Cuando el sistema obtiene una lista temporal de artículos elegidos.

Flujo normal:
1. El usuario pulsa en la opción "catálogo".
2. El sistema muestra el catálogo.
3. El sistema muestra las categorías existentes.
4. El usuario elige la categoría que más le convenga.
5. El sistema muestra las subcategorías correspondientes.
6. El usuario elige la subcategoría que desea.
7. El sistema muestra la lista de los artículos.
8. El usuario elige el(los) artículo(s).
9. El sistema le muestra una lista de los artículos elegidos (incluyendo nombres, marcas, cantidades, precios unitarios, subtotal, IVA y total).
10. El usuario revisa la lista y confirma su decisión pulsando "confirmar lista".
11. El sistema guarda una lista temporal.
Flujos de excepción:
1. No hay subcategorías. Mostrar artículos.
2. Al ver la lista finalizada, el usuario se da cuenta de que le falta añadir artículos a la misma. El usuario pulsa "añadir" y el sistema muestra la página de catálogo en la cual podrá seguir escogiendo artículos.
3. Al ver la lista finalizada, el usuario ya no desea algunos productos. El usuario pulsa "quitar" y el sistema muestra una copia de la lista en donde el usuario podrá elegir los productos que desea quitar.
4. Al ver el monto total de la lista, el usuario se da cuenta de que no cuenta con el suficiente crédito para pagar esa suma. El usuario pulsa "cancelar" y el sistema regresa al usuario a la página principal.
5. El monto total es menor a $500.00. El sistema le mandará un mensaje de error, indicándole que la cantidad mínima para realizar un pedido es de $500.00 y lo regresará al catálogo.
6. El cliente no ha terminado de pagar su pedido anterior en la fecha indicada por el sistema. El sistema le mandará un mensaje indicándole que tiene adeudos actualmente y por lo tanto no puede solicitar el pedido hasta que no cubra con su pago.
7. El usuario pulsa en la opción "cancelar". El sistema muestra un mensaje en el que se lee que ha optado por no elegir ningún artículo y lo envía a la página principal.

8. El usuario no elige artículo. Cerrar cesión después de un determinado tiempo.

Caso de uso #13.

Nombre: "Elegir forma de pago".
Precondiciones: El usuario deberá tener una lista de artículos elegidos (enviada por el sistema).
Poscondiciones: El usuario habrá elegido una forma de pago.
Inicio: Cuando el sistema le pide al cliente defina una forma de pago.
Fin: Cuando el sistema confirme la forma de pago del cliente.
Flujo normal:
1. El usuario pulsa en la opción "elegir forma de pago".
2. El sistema muestra dos opciones de tipo de pago (de contado o a crédito).
3. El usuario escoge una de ellas.
4. El sistema muestra las formas de pago para cada uno de los tipos de pago (efectivo, cheque, o transacción bancaria).
5. El usuario elige la forma de pago que desea.
6. El sistema muestra el tipo y su respectiva forma de pago.
7. El usuario confirma su decisión.
Flujos de excepción:
1. El usuario está realizando compras por primera vez y escoge pagar a crédito. Se muestra un error comunicándole al cliente que "por políticas de la empresa las primeras compras de un cliente deberán ser de contado y posteriormente se podrá comenzar a valorar créditos" y se le invita a escoger otro tipo de pago.
2. El usuario pulsa en la opción "cancelar". El sistema muestra un mensaje en el que se lee que no ha optado por ninguna forma de pago, por lo tanto se ha cancelado su lista de artículos y lo envía a la página principal.
3. El usuario no elige ninguna forma de pago. Cerrar cesión después de un determinado tiempo.

Caso de uso #14.

Nombre: "Solicitar el pedido".
Precondiciones: El usuario deberá haber elegido una forma de pago.
Poscondiciones: El usuario habrá solicitado el pedido.
Inicio: Al solicitar un pedido.
Fin: Al recibir la confirmación de la solicitud del pedido por parte del sistema.

Flujo normal:
1. El usuario pulsa en la opción "solicitar pedido".
2. El sistema envía una respuesta al usuario que su pedido está en proceso.
Flujos de excepción:
1. El sistema muestra un mensaje, en el cuál se lee que hubo algún problema al enviar la solicitud de pedido. El sistema le sugiere repetir el proceso y es enviado al catálogo.
2. El usuario pulsa en la opción "cancelar". El sistema muestra un mensaje en el que se lee que ha optado por cancelar su solicitud de pedido y lo envía a la página principal.

Caso de uso #15.

Nombre: "Solicitar cancelación de un pedido por parte del cliente".
Precondiciones: El usuario deberá haber solicitado un pedido.
Poscondiciones: El usuario habrá cancelado el pedido.
Inicio: Al solicitar la cancelación de un pedido.
Fin: Al recibir la confirmación de la cancelación del pedido por parte del sistema.
Flujo normal:
1. El usuario pulsa en la opción "cancelar pedido".
2. El sistema envía al usuario a un formulario de cancelación.
3. El usuario llena el formulario.
4. El usuario pulsa en la opción de "solicitar cancelación de pedido".
5. El sistema envía una respuesta al usuario que su pedido está siendo analizado para aprobar su cancelación.
6. El sistema aprueba la cancelación y efectúa los cambios en la base de datos.
7. El sistema le confirma al cliente la cancelación del pedido.
Flujos de excepción:
1. El usuario llenó algunos campos incorrectamente. Se manda un mensaje de error comunicándoselo e indicándole cuales son los campos en los que se ha equivocado, para que pueda corregirlos.
2. El usuario no llenó los campos obligatorios. Se manda un mensaje de error comunicándoselo e indicándole cuales son los campos que son necesarios para la cancelación.
3. El envío del formulario no ha sido satisfactorio. Se manda un mensaje de error

comunicándoselo y se le invita a volver a mandarlo.
4. El pedido solicitado para cancelación no existe. El sistema manda un mensaje de error comunicándoselo.
5. El usuario no llena los campos. Cerrar cesión después de un determinado tiempo.
6. El usuario no pulsa "solicitar cancelación de pedido". Cerrar cesión después de un determinado tiempo.
7. El sistema no aprueba la cancelación. El sistema manda un mensaje en el que le indica al cliente que la solicitud de cancelación de su pedido no ha sido aprobada y le indicará en que estado se encuentra (por ejemplo, en camino para ser entregado) y lo mandará a la página principal.
8. El usuario pulsa en la opción "cancelar". El sistema muestra un mensaje en el que se lee que ha optado por cancelar su solicitud de cancelación de pedido y lo envía a la página principal

Caso de uso #16.

Nombre: "Surtir el pedido".
Precondiciones: El usuario deberá haber solicitado el pedido.
Poscondiciones: El pedido estará completo para ser entregado.
Inicio: Al enviar la solicitud de un pedido.
Fin: Al tener completo el pedido.
Flujo normal:
1. El sistema compara las existencias de los artículos solicitados por el cliente.
2. El sistema enviará la solicitud del pedido (lista del cliente) al administrador. Además mostrará en esta lista el número de artículos con los que se contaba y el número de artículos con los que se cuenta actualmente en el almacén por añadirlos en este pedido.
3. El administrador revisa la solicitud del pedido y confirma el pedido.
4. El sistema crea un nuevo pedido con todos sus datos (cliente, artículos, cantidad, precio unitario, subtotal, IVA, total, fecha de solicitud, fecha de confirmación, etc.). 5. El pedido está completo para ser entregado.
Flujos de excepción:
1. No hay suficientes existencias de uno o más artículos en el almacén para surtir

el pedido. El sistema muestra los artículos insuficientes en color rojo, además de un mensaje en el cuál se le indica al administrador que tiene existencias insuficientes de uno o más productos para completar el pedido. El administrador debe adquirir los artículos que hacen falta con su proveedor.
2. Al mostrar el sistema el número de artículos con los que se contará en el almacén, el administrador puede ver que las cantidades que le quedarán serán mínimas. El administrador debe adquirir más artículos con su proveedor para que pueda seguir surtiendo pedidos.

Caso de uso #17.

Nombre: "Realizar la factura".
Precondiciones: El administrador deberá haber confirmado el pedido.
Poscondiciones: El sistema habrá realizado la factura.
Inicio: Cuando el administrador del sistema confirma un pedido.
Fin: Cuando el sistema realiza la factura.
Flujo normal:
1. El administrador pulsa en la opción "realizar factura".
2. El sistema añade automáticamente los datos del cliente, de su pedido y las condiciones de pago al formato de la factura.
3. El administrador revisa los datos de la factura y la confirma.
4. El sistema realiza e imprime la factura.
Flujos de excepción:
1. Los datos de la factura son erróneos. El sistema muestra al administrador la opción de corregir los datos de la factura.
2. La impresora no tiene papel. El sistema muestra un mensaje, en el cuál le indica al administrador que debe colocar más papel en la impresora.

Caso de uso #18.

Nombre: "Entregar pedido".
Precondiciones: El administrador habrá surtido completamente el pedido y el sistema deberá haber impreso la factura.
Poscondiciones: El pedido aparecerá en el sistema como "entregado".
Inicio: Cuando el administrador completa un pedido y el sistema imprime la factura.
Fin: Cuando el pedido es entregado.

Flujo normal:
1. El pedido es enviado al domicilio de entrega del cliente.
2. El cliente revisa el pedido y confirma que esté completo.
3. El pedido es entregado al cliente.
Flujos de excepción:
1. El domicilio al que fue enviado no existe. El administrador se comunica con el cliente para confirmar el domicilio y se actualizan los datos en la base de datos.
2. El pedido no está completo. El cliente no recibe el pedido hasta que no se le entregue completo.
3. Los datos de la factura son erróneos. El cliente no recibe el pedido, ni tampoco la factura. Se procede a corregir los datos de la factura, para poder entregar el pedido y la factura al cliente.

Caso de uso #19.

Nombre: "Cobrar pedido".
Precondiciones: El cliente habrá recibido el pedido y la factura.
Poscondiciones: El pedido aparecerá en el sistema como "cobrado".
Inicio: Cuando el cliente recibe el pedido y la factura.
Fin: Cuando el pedido es cobrado.
Flujo normal:
1. El cliente procede a pagar su pedido (en las condiciones de pago establecidas).
a) En efectivo. El cliente paga a la persona que le entregó el pedido y la empresa deposita el dinero en su cuenta.
b) Por cheque. El cliente da el cheque a la persona que le entregó el pedido y la empresa procede a cobrarlo en el banco.
c) Por transacción bancaria. El cliente realiza la transacción bancaria y después la empresa revisa su estado de cuenta.
2. El sistema actualiza el campo del pedido como "cobrado".
Flujos de excepción:
1. El cheque no tiene fondos. La empresa se comunica con el cliente o bien lo localiza personalmente para que pueda realizar su pago satisfactoriamente.
2. La transacción bancaria no ha sido realizada. La empresa se comunica con el cliente o bien lo localiza personalmente para que pueda realizar su pago satisfactoriamente.

3. Requerimientos no funcionales

Se requiere diseñar la infraestructura de un sistema que permita hacer más eficiente el servicio de solicitud y realización de pedidos de la empresa, mediante la adecuada administración y control de existencias de los artículos, así como de los diferentes tipos de clientes, para obtener el máximo aprovechamiento de tiempo y servicio.

6.1 Requerimientos del medio ambiente

Servidor:	Cliente:
Procesador Celeron a 895 Mhz. 256 MB memoria RAM. Disco Duro 14 GB. Máquina Virtual de Java JDK 1.2.x o superior. API JDBC 2.0 Servidor Apache Servidor Jakarta Tomcat Acceso a Base de Datos DB2 o MySQL	PC con Acceso a Internet Máquina Virtual de Java

CAPITULO 3. LENGUAJE DE PROGRAMACIÓN JAVA PARA LA CREACIÓN DE UNA APLICACIÓN WEB

3.1 Introducción a Java.

Java fue creado en 1990 por James Gosling, de *Sun Microsystems*[1], como un software para dispositivos electrónicos como calculadoras, microondas y televisión Interactiva e inicialmente se llamó Oak (roble). Las razones por las que se creó Java fueron: la creciente necesidad de interfaces más cómodas e intuitivas, fiabilidad del código, facilidad de desarrollo y la enorme diversidad de controladores electrónicos. Finalmente en agosto 1995 Java fue presentado como lenguaje de programación enfocado a Internet.

Java es, tanto un lenguaje, como una plataforma de software desarrollada por *Sun Microsystems*. Ésta plataforma ha sido desarrollada de tal manera que los programas desarrollados para ella puedan ejecutarse de la misma forma en diferentes tipos de arquitecturas y dispositivos computacionales.

3.1.1 Características de Java

Java es entonces un lenguaje de programación, y tiene las siguientes características:

→ Es simple, pues no posee aritmética de punteros, no se necesitan sentencias para borrar (delete), y no tiene herencia múltiple. Es similar a C++, y no se requiere de mucha capacitación para poder programar

→ Es Orientado a objetos, ya que maneja objetos, clases, métodos, subclases, herencia simple y encapsulamiento, que son características estándares de los lenguajes orientados a objetos.

→ Es distribuido, ya que se ha construido con capacidades de interconexión TCP/IP, pues tiene librerías de rutinas para acceder e interactuar con protocolos como HTTP y FTP.

→ Es tanto interpretado como compilado, es decir, que el código fuente es compilado creando un byte-code, que es un código intermedio de muy bajo nivel, pero sin alcanzar el código máquina, por lo tanto este mismo código puede ser ejecutado en diferentes equipos con cualquier plataforma. Para lo que es necesario

[1] http:// java.sun.com

interpretarlo, esto lo hace la maquina virtual de java, que es la que interpreta éste código.

→ Es robusto, ya que hace chequeo de índices en arreglos, chequeo de tipos durante la ejecución, contiene un recolector de basura (Garbage Collector), que administra la memoria y no posee aritmética de punteros. Tiene un modelo de manejo de memoria que evita los apuntadores de C++.

→ Es seguro, porqué no se puede acceder a la memoria libre, utiliza el método de autentificación de llave pública y además aplica restricciones a programas de red, archivos y aplicaciones nativas, es decir, que las aplicaciones desarrolladas en Java y ejecutadas vía Internet no pueden ser modificadas sin autorización.

→ Es Multiplataforma, esto significa que es independiente de la arquitectura, pues el compilador Java genera "Java bytecode" el cual es un formato independiente a cualquier arquitectura, diseñado para transportar código entre diferentes plataformas de hardware y software, por esto posee compatibilidad con cualquier plataforma, sin necesidad de hacer ningún cambio, a nivel de fuentes, de bibliotecas y también a nivel de código compilado.

→ Es portátil porque puede ejecutarse en diferentes plataformas, ya que cuenta con una "Java Virtual Machine"[2], o máquina virtual de java que interpreta el código java.

→ Es Multithread, esto es, que puede realizar muchas actividades simultáneas en un programa.

→ Es dinámico, porqué las aplicaciones son adaptables a cambios en el ambiente, pudiéndose cargar nuevos módulos desde cualquier sitio de la red.

3.1.2 Tipos de aplicaciones en Java

Standalone: Son los programas básicos de Java, que se ejecutan por el Java Runtime Enviroment (JRE). No necesitan la red para ejecutarse, contienen un método principal llamado ***main.***

Applets: Son mini aplicaciones de java, éstos programas se deben incluir en páginas web para poder ser observados por otra aplicación y se ejecutan en el navegador web cuando el usuario carga la página Web. Deben incluir un método de nombre

[2] Máquina Virtual de Java, encargada de ejecutar el código generado por el compilador de Java.

start(), que será ejecutado cuando el navegador cargue la página Web. Éstas aplicaciones son seguras y no permiten incluir métodos nativos de Java.

Servlets: Son módulos que permiten sustituir o utilizar el lenguaje Java en lugar de programas CGI (Common Gateway Interface)[3]. Éstos se ejecutan como aplicaciones en servidores en Internet, no tienen entorno gráfico, pues su respuesta es código HTML.

JSP's: Contienen elementos HTML y código Java, se podría decir que son un tipo de abstracción de un Servlet.

EJB's: Es un componente que permite agrupar funcionalidades para formar parte de la aplicación, es decir, es un componente de despliegue que a través de un EJB Container o contenedor EJB proporciona los siguientes servicios y ventajas: Servicios Middleware[4], División de Trabajo, Diversos Vendedores, Procedimientos Remotos (RMI) y Diversos Clientes.

JavaBeans: Es un componente que permite agrupar funcionalidades como parte de una aplicación y permite reutilizar componentes de software. Un JavaBean contiene los métodos ***get()*** y ***set()***.

3.2 Recomendaciones de programación

→ Utilizar nombres significativos para los identificadores y variables, y respetar la siguiente notación de mayúsculas y minúsculas:
- En las clases: Clase o MiClase.
- En las interfaces: Interfaz o MiInterfaz.
- En los métodos: método() o métodoLargo().
- En los métodos de acceso: getAtributo(), setAtributo().
- En las variables: altura o MiAltura.
- En las constantes: CONSTANTE o CONSTANTE_LARGA.
- En los paquetes: java.paquete.subpaquete.

→ Cuidar los nombres de las clases y paquetes para que no coincidan con otros ya existentes, mediante la utilización de prefijos identificativos.

→ No utilizar líneas de más de 80 caracteres, en vez de esto, utilizar varias líneas.

[3] CGI, son programas que tienen una pantalla común desde la cual se ejecutan todas las acciones.
[4] Software de comunicaciones que reside físicamente en el cliente remoto y en un servidor de comunicaciones, localizado entre el cliente y el servidor de aplicaciones. Es el software que actúa como un traductor universal entre distintas tecnologías de radiofrecuencia y protocolos.

→ Escribir una sola operación por cada línea.
→ Insertar líneas en blanco en el código fuente para mejorar la legibilidad.
→ Incluir comentarios siempre en las clases, describiendo su funcionalidad.

3.3 Sintaxis del Lenguaje.

3.3.1 Comentarios:

Existen 3 tipos de comentarios que son: // para una sola línea, /* para más de una línea / y /** de documentación, de una o más líneas **/

3.3.2 Identificadores:

Los identificadores nombran variables, funciones, clases y objetos, un identificador comienza con una letra, un subrayado(_) o un símbolo de dólar($), los siguientes caracteres pueden ser letras o números. Los identificadores distinguen las mayúsculas de las minúsculas y no existe una longitud máxima.

3.3.3 Palabras clave

Las siguientes son las palabras clave que están definidas en Java y que no se pueden utilizar como identificadores:

abstract	continue	for	new	switch
boolean	default	goto	null	synchronized
break	do	if	package	this
byte	double	implements	private	threadsafe
byvalue	else	import	protected	throw
case	extends	instanceof	public	transient
catch	false	int	return	true
char	final	interface	short	try
class	finally	long	static	void
const	float	native	super	while

3.3.4 Palabras Reservadas

Además, el lenguaje se reserva unas cuantas palabras más, pero que hasta ahora no tienen un cometido específico. Son:

cast	future	generic	Inner
operator	outer	rest	var

3.3.5 Variables

Son objetos que almacenan sus valores en variables. Existen dos tipos de datos para una variable, que son: primitivos y por referencia. Una variable de tipo primitivo contiene un único valor de tamaño y formato apropiado para su tipo. Los arreglos, las clases y las interfaces son variables de tipo por referencia, esto es, que su valor una referencia a la dirección donde se localiza el valor o conjunto de valores representados por la variable.

3.3.6 Operadores

Un operador es el que ejecuta una función entre 1, 2 ó 3 operandos.

Los tipos de operadores de java son:

→ Operadores aritméticos: +, -, *, /, %, ++, –
→ Operadores relacionales: >,<, <=, >=, ==
→ Operadores lógicos: &&, ||, !, &, |, ^
→ Operadores de asignación: =

3.3.7 Estructuras de control

Las estructuras de control permiten controlar el flujo del programa y son:

3.3.7.1 Condicionales

→ **If/else**: Ésta sentencia proporciona a los programas la posibilidad de ejecutar selectivamente otras sentencias basándose en algún criterio.

→ **switch:** Se utiliza para realizar sentencias basadas en alguna expresión.

3.3.7.2 Ciclo

→ **for**: sentencia que se utiliza frecuentemente para crear ciclos.

→ **while**: Se realiza una acción mientras que se cumpla una condición.

→ **do/while**: Es similar a ***while***, pero ésta sentencia se realiza al menos una vez dentro del programa, porqué la condición se evalúa al final.

3.3.7.3 Manejo de excepciones:

Las excepciones son la manera que ofrece Java de manejar los errores en tiempo de ejecución. Las excepciones nos permiten escribir código que nos permita manejar ese error y continuar con la ejecución del programa.

3.3.7.4 Control general de flujo

→ break [etiqueta]
La sentencia break señala el fin o rompimiento del ciclo for, while, etc., que correspondan a la etiqueta.

→ Continue [etiqueta]
La sentencia continue, indica la secuencia que debe seguir el programa.

→ return expresión;
La sentencia return devuelve al método el valor de la expresión o variable, también se puede utilizar para salir del método sin necesidad de devolver nada.

→ etiqueta: sentencia;
Las etiquetas se utilizan para identificación, cuando existen varios ciclos anidados.

3.3.8 Clases

Una clase es un prototipo, que define las variables y los métodos comunes a un cierto tipo de objetos. Es decir, las clases son las plantillas de las que se pueden crear múltiples objetos del mismo tipo. La clase define las variables y los métodos comunes a los objetos de ese tipo, pero cada objeto, a su vez, tendrá sus propios valores y compartirán las mismas funciones. Primero se debe crear la clase antes de poder crear objetos o ejemplares de esa clase. Una clase consta de 2 partes la declaración y el cuerpo de la clase. La forma en que declara una clase es la siguiente:

public / abstract / final class *NombreClase* **extends Super/implements interfaces**

{ *CuerpoDeClase* }

3.3.9 Arreglos

En Java un arreglo es realmente un objeto, porque tiene predefinido el operador [ ]. Los límites de los arrays (arreglos) se comprueban en tiempo de ejecución para evitar

desbordamientos y la corrupción de memoria. Para crear un array en Java hay dos métodos básicos, que son:
Crear un array vacío, de la siguiente forma: ***int lista[ ] = new int[50];***
Crear el array con sus valores iniciales:
String nombres[ ] = {"Juan","Pepe","Pedro","Maria"};

3.3.10 Métodos

Son funciones que pueden ser llamadas dentro de la clase o por otras clases. Los métodos están formados por 2 partes: la declaración y el cuerpo o contenido. La declaración del método define todos los atributos del método, como son nivel de acceso, tipo de retorno, nombre del método y parámetros. Una declaración debe contener, por lo menos, el tipo de retorno y el nombre del método. Dentro del cuerpo del método se especifican todas las instrucciones que realiza el método.

3.3.11 Objetos

Un objeto es un conjunto de variables y métodos relacionados con esas variables, es decir, un objeto contiene en sí mismo la información y los métodos o funciones necesarios para manipular esa información.

Los objetos se crean a partir de las clases. Para crear y utilizar objetos se deben declarar, instanciar e inicializar. Primero se crean como una clase nueva, que contenga los atributos del objeto, después se debe declara el objeto y luego crear una instancia e inicializa, utilizando el operador **new** (p. ej. cosa = *new* objeto(silla);).

3.4 Ejemplo para la Programación de la Aplicación del Proyecto

3.4.1 Componentes

De presentación

- Gestión del control de acceso
- Generación de usuario/contraseña para público en general
- Carrito de compras
- Catalogo de productos

Del Negocio

- Producto
- Pedido
- Entrega

- Cobro
- Factura
- Cliente
- Catalogo de productos

Componentes de Datos

- Recuperar contraseña del usuario
- Crear acceso a usuario
- Listar productos de catalogo
- Consultar lista de pedidos

3.4.2 Programas Básicos para el manejo de una Base de Datos

// Programa que crea una Tabla

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class crearTabla {
   public static void main(String[] args)
   {  try
     {  // Cargando el  controlador para DB2
        Class.forName("com.ibm.db2.jcc.DB2Driver");
        // estableciendo  una conexión para DB2
        Connection db2Conn = DriverManager.getConnection
("jdbc:db2://132.248.44.94:50000/diploweb","db2web","db2admin");
        Statement st = db2Conn.createStatement();
        String myUpdate = "CREATE TABLE
db2web.ravc_DATOSPERSONA(IdPersona Integer, calle VARCHAR(50),numero
VARCHAR(20),colonia VARCHAR(80),idEstado Integer,cp VARCHAR(10))";
        st.executeUpdate(myUpdate);
        st.close();
        db2Conn.close();
     } // fin try
```

```
    catch (ClassNotFoundException cnfe)
    {   cnfe.printStackTrace();      } // fin catch
    catch (SQLException sqle)
    {   sqle.printStackTrace();
    } // fin catch
  } // fin main
} // fin de programa
```

// Programa que sirve para introducir datos en la Tabla creada

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class usarTabla {
  public static void main(String[] args)
  {     try
    {    // cargando el controlador para DB2
        Class.forName("com.ibm.db2.jcc.DB2Driver");
        // estableciendo  una conexión para DB2
        Connection db2Conn =
      DriverManager.getConnection
       ("jdbc:db2://132.248.44.94:50000/diploweb","db2web","db2admin");
        Statement st = db2Conn.createStatement();
        String myUpdate = "INSERT INTO db2web.ravc_DATOSPERSONA
       VALUES(6,'Hacienda','369','Bosques de Aragon ',1,'54071')";
        st.executeUpdate(myUpdate);
        st.close();
       db2Conn.close();
  } // fin try
    catch (ClassNotFoundException cnfe)
    { cnfe.printStackTrace();    } // fin catch
    catch (SQLException sqle)
    { sqle.printStackTrace();      } // fin catch
  } //fin main
} // fin programa
```

```
// Programa para Consultar los datos de la Tabla
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class consultaTabla {
   public static void main(String[] args)
   {    try
     {        // cargando el Driver de DB2
        Class.forName("com.ibm.db2.jcc.DB2Driver");
        // estableciendo una conexión a DB2
        Connection db2Conn = DriverManager.getConnection
        ("jdbc:db2://132.248.44.94:50000/diploweb","db2web","db2admin");
        Statement st = db2Conn.createStatement();
        String myQuery = "SELECT * FROM db2web.ravc_DATOSPERSONA";
        ResultSet resultSet = st.executeQuery(myQuery);
        // cycle through the resulSet and display what was grabbed
        while (resultSet.next())
        {  Integer id = Integer.valueOf(resultSet.getString("IdPersona"));
           String  calle = resultSet.getString("calle");
           String Number = resultSet.getString("numero");
           String colonia = resultSet.getString("colonia");
           Integer est = Integer.valueOf(resultSet.getString("IdEstado"));
           String cp = resultSet.getString("cp");
           System.out.println("Id: " + id);
           System.out.println("calle: " + calle);
           System.out.println("numero: " + Number);
           System.out.println("colonia: " + colonia);
           System.out.println("estado: " + est);
           System.out.println("codigo: " + cp);
           System.out.println("----------");
        } // fin ciclo while
           resultSet.close();
        st.close();
```

```
        db2Conn.close();
      } // fin try
      catch (ClassNotFoundException cnfe)
      {       cnfe.printStackTrace();     } // fin catch
      catch (SQLException sqle)
      {       sqle.printStackTrace();     } // fin catch
    } // fin main
} //fin de programa
```

// Programa para actualizar los datos de la Tabla

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class updateTabla {
   public static void main(String[] args)
   {    try
     { // cargando el controlador para DB2
        Class.forName("com.ibm.db2.jcc.DB2Driver");
        // estableciendo  una conexión para DB2
        Connection db2Conn =  DriverManager.getConnection
("jdbc:db2://132.248.44.94:50000/diploweb","db2web","db2admin");
        Statement st = db2Conn.createStatement();
        String myUpdate = "DELETE FROM db2web.ravc_DATOSPERSONA WHERE
IdPersona=6";
        st.executeUpdate(myUpdate);
        st.close();
       db2Conn.close();     } //fin try
      catch (ClassNotFoundException cnfe)
      {       cnfe.printStackTrace();     } // fin catch
      catch (SQLException sqle)
      {       sqle.printStackTrace();     } // fin catch
    } // fin main
} // fin de programa
```

CAPITULO 4. REGLAS PARA EL DISEÑO DE LA INTERFAZ DE USUARIO

4.1 Interfaz de usuario

Es donde se realiza la presentación de datos a través de una pantalla y posee las siguientes características:

- El usuario interactúa con la información, en el cliente.
- La información se despliega en el navegador, y es presentada como un documento HTML.
- El HTML es un lenguaje para presentar documentos multimedia y acceder a recursos disponibles en Internet.

La interfaz es el medio por el cual el usuario puede hacer todas sus operaciones a la base de datos, a través de operaciones básicas de Internet, por lo que en su diseño se debe

- Identificar los elementos con que se constituirá o formará: textos, imágenes, colores, referencias, audio, ligas, las partes de la página.
- Definir el aspecto visual.
- Definir el contenido informativo.
- Definir su estructura o la organización de: los contenidos, la información y los nexos o ligas.

4.2 Proceso de diseño

El diseño de la interfaz se basa en los procesos y tareas de los usuarios, para esto es necesario seguir los siguientes pasos:

a. Entender quien usará el sistema y para que.

Es un **"análisis de usuarios y tareas"**, donde se obtiene la información de la definición de casos de uso y perfiles de usuario, esto es, que se identifican los niveles de usuario que el sistema va a tener y el nivel de acceso.

Los niveles de usuario sirven para establecer los tipos de acceso y operaciones que pueden realizar los diferentes usuarios, para lo cual se establecen las cuentas correspondientes, pudiendo ser de:

1) Sólo consulta básica o de área.
2) Consulta total.

3) Consulta y operaciones básicas o de área.
4) Realización de operaciones totales.
5) Consulta, operaciones y generación de reporte de área.
6) Acceso total.

En esta fase del diseño se desarrolla el Mapa de navegación general, el cual se usa para verificar cuales son los procesos u operaciones que cada nivel de usuario tiene permitido realizar y además Identificar los procesos y/o secciones que conformaran la interfaz del Sistema Web, los cuales se complementarán de manera específica con los diagramas de secuencia.

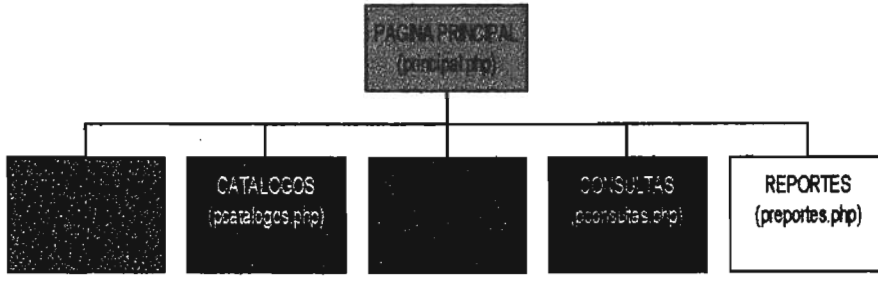


Fig.1 Mapa de Navegación

b. Utilizar un estándar o normatividad de la empresa u organización.

Si se cuenta con un estándar para el diseño de la interfaz es necesario ajustarse lo más posible a éste, o en caso contrario justificar porque no será así.

Es frecuente que se tengan sistemas de información o sistemas Web en producción, en donde ya se tienen definidos los colores, tamaños de letras, logotipos, frases, etc. por lo tanto se pueden analizar interfaces existentes con las cuales los usuarios han trabajado y después tomar y construir ideas para posteriormente utilizarlas en la nueva interfaz.

c. Bosquejar un diseño.

La descripción del bosquejo del diseño debe ser puesto en papel, el cuál fuerza a que se piense acerca de los elementos de las pantallas y el orden que éstos tendrán, aunque la información se toma del análisis y los diagramas en ocasiones dada la gran cantidad de elementos se pueden tener omisiones si se decide programar directamente.

d. Crear un prototipo.

Después de hacer el bosquejo y comprobar que se están considerando todos los requerimientos, si es necesario se debe implementar un prototipo que se presente a los usuarios, obviamente sin conexiones a la base de datos, para comprobar la congruencia de la interfaz con los requerimientos del usuario, aunque es un trabajo más detallado, en ocasiones puede ayudar a encontrar omisiones y un orden inadecuado.

e. Evaluación con los usuarios.

Existen muchos problemas que solamente aparecerán cuando el diseño se está probando con los usuarios. La evaluación debe de ser hecha con la gente cuyo conocimiento y expectaciones se acerquen más a los usuarios reales. Los usuarios deben de ser cuestionados sobre el rendimiento de una o más tareas representativas que han sido diseñadas en el sistema para su mantenimiento. Ellos deben de ser cuestionados en la técnica de "pensamiento en voz alta" la cual consiste en que el usuario vaya narrando todo lo que está pensando al momento de realizar una tarea.

f. Construcción.

Una vez que se ha comprobado la congruencia del diseño con los usuarios se puede empezar a construir la interfaz. Trate de anticipar cambios menores o secundarios con sencillos cambios de variables.

4.3 Criterios de usabilidad.

- **Diseño amigable**: Toda interfaz debe ser amigable, pero más en sistemas Web, ya que el usuario la visualiza como un elemento de Internet, aunque la funcionalidad del sistema lo puede hacer olvidar lo vistoso de las páginas comerciales, siendo fundamentales el diseño y formato.
- **Congruencia con los procesos**: El sistema debe estar apegado a los procesos establecidos en el análisis, siendo fundamental su funcionalidad y organización, una interfaz amigable no sirve de nada si los procesos no son los necesarios ni adecuados.
- **Distribución lógica**: Los elementos de la interfaz como formularios, botones, listas, tablas, celdas, imágenes, mensajes, etc. Deben tener una distribución lógica para lograr la funcionalidad y entendimiento.

- **Navegación lógica**: El orden y secuencia en que se navegue entre los elementos de la interfaz y sus pantallas deben ser adecuados y lógicos según los procesos y perfiles de usuario.
- **Retroalimentación**: Es la información o respuesta que da el sistema a través de mensajes, a las acciones o procesos que realiza el usuario. Es conveniente el uso de mensajes para indicarle al usuario que se ésta procesando o cargando información y que no es un problema o error de conexión.
- **Consistencia**: La uniformidad de la distribución, navegación, diseño, entradas y salidas, uso de colores, formato de mensajes, etc. de todas las pantallas debe conservarse siempre, pues el usuario se adaptará rápidamente al uso del sistema con un comportamiento homogéneo de éste.
- **Implementación de ayuda:** Es necesaria la implementación de módulos de ayuda, esto se decide a través del análisis de los perfiles de usuario.

4.4 Estructura y organización de las páginas de la interfaz.

El equilibrio entre la estructura y las relaciones entre las páginas y sus procesos es importante pues el objetivo es construir una jerarquía de menús y páginas que parezcan normales y bien estructuradas al usuario.

Los modelos tradicionales son: lineales, planos, jerárquicos, vertical y vertical con opciones, sin embargo en Sistemas Web, la estructura la dan los procesos y generalmente es del tipo vertical con opciones.

La estructura de las páginas debe ser clara, funcional y con una continuidad gráfica entre los diferentes componentes y subsecciones del Sistema Web, de manera que la relación entre componentes sea ordenada.

4.5 HTML

HTML es el lenguaje más conocido para la creación de páginas web, es un lenguaje descriptivo muy sencillo y por su integración con HTTP, combina la capacidad de HTTP para enviar y recibir documentos HTML y se logra un protocolo de comunicaciones, que aunque es muy vulnerable, admite una gran variedad de protocolos de autenticación soportados por HTML como:

a. Básico. Nombre de usuario/contraseña en claro, codificado en Base-64.
b. Resumen o Digest. la contraseña no se codifica en claro.

c. Basado en formularios. Se utiliza un formulario personalizado para introducir el usuario y contraseña.
d. NTLM. Es el protocolo de autenticación de Microsoft, implementado dentro de las cabeceras solicitud/respuesta de HTTP.
e. Negociado. Es un protocolo de Microsoft que permite cualquier tipo de autenticación definida para poner de acuerdo dinámicamente al cliente y al servidor.
f. Certificado del lado del cliente. Aunque se utiliza en muy pocos casos, SSL/TLS proporciona la posibilidad de comprobar la autenticidad de un certificado digital presentado por el cliente Web.

La principal desventaja de HTML es no poder generar páginas dinámicas, sin embargo esta desventaja puede ser solucionada al utilizar PHP o Java, para volver la dinámica, por lo que dentro del código HTML se puede contener otra herramienta ya que permita el uso de:

→ Servlets. Extienden las capacidades de los servidores web.
→ Applets. Extienden las capacidades del lado del cliente.
→ Middlets. Extienden las capacidades de los microdispositivos.

4.5.1 Estructura de un documento HTML

Un documento HTML es un conjunto de diversos datos que son interpretados por un programa visualizador, no es un lenguaje e programación, es denominando como un lenguaje de etiquetado, su estructura consta de dos partes: cabeza o encabezado y cuerpo.
El encabezado incluye datos relacionados con el contenido del documento, como el titulo del mismo, el autor, y en algunos casos palabras clave sobre el contenido de la página, también puede contener datos o tags (etiquetas).
El cuerpo del documento HTML contiene la información que pretende suministrar dicho documento, texto, imágenes, etc.
Un documento HTML es un documento de texto sin formato, que contiene instrucciones llamadas tag o etiquetas que un visualizador interpreta, por medio de un código fuente, las cuales tienen el siguiente formato:

<tag> *contenido* **</tag>**

4.5.2 Componentes principales de un documento HTML

4.5.2.1 Tag <HTML>.

El tag <HTML>, se define como el tag principal, pues se incluye al principio de cualquier cosa dentro de un documento HTML, este tag va acompañado del tag </HTML> que contiene una diagonal que indica que la función de este tag es finalizar la acción del primero, el tag **</HTML>** se coloca al final de cualquier dato o tag que pudiese contener el documento.

4.5.2.2 Tag <HEAD>

El tag <head> se usa para definir el encabezado de un documento HTML. **todo lo que se coloque después del tag** <HEAD> **y antes del tag** </HEAD> se considera la cabeza, este tag se cierra con </HEAD>.

4.5.2.3 Tag <BODY>.

La etiqueta <BODY> se usa para definir el cuerpo de un documento HTML. Todo lo que se coloque después del tag <BODY> y que éste antes del tag </BODY> se considera el cuerpo del documento.

Las etiquetas pueden tener atributos, los cuales sirven para darle formato y estilo a los contenidos y cuerpo de los documentos, por ejemplo: color, tamaño, ubicación, etc.

4.5.2.4 Listas.

Las listas en una interfaz son esenciales para el manejo de la información, pues permiten crear texto con varios formatos de listado, las cuales pueden ser:

- **Ordenadas** se refiere a numerados, no ordenados por algún criterio.
- **Desordenadas** no numerados.
- **Directorios.**
- **Menú.**
- **Listados de definición.**

4.5.2.5 Tablas.

Las tablas son elementos prácticos para ordenar información dentro de celdas, que pueden ser localizadas por medio del número de columna y renglón en que se encuentra.

La etiqueta **<TABLE>** nos permite crear una tabla, todo lo que este dentro de estas etiquetas será considerado como parte de la tabla, la etiqueta de cierre es **</TABLE>**, dentro se utilizan las etiquetas **<TR>** (renglones)y **<TD>** (celdas) su cierre es opcional.

4.5.2.6 Imágenes.

El formato de imágenes más utilizado es el **GIF** para logotipos, pero el **JPEG** es el más adecuado para fotos.

4.5.2.7 Hipervínculos.

Los hipervínculos son un elemento necesario en la interfaz, pues permite acceder a otros módulos, páginas HTML, imágenes y URL´s, para lo cual se utilizan las siguientes etiquetas:

4.5.2.8 Formularios.

Son los elementos fundamentales para introducir información para su elaboración se utilizan las etiquetas siguientes:

- **<FORM>** que define un formulario con diversos elementos.
- **ACTION**, que especifica la acción a tomar después de enviar.
- **METHOD** indica la forma en que se enviarán los datos del formulario, pude ser **POST** o **GET**, pero el más seguro es post, porqué no muestra los parámetros.
- **<INPUT>** permite crear los elementos que se necesitan utilizar dentro del formulario.

4.6 Interfaz de proyecto

4.6.1 Pantallas Principales del Sistema

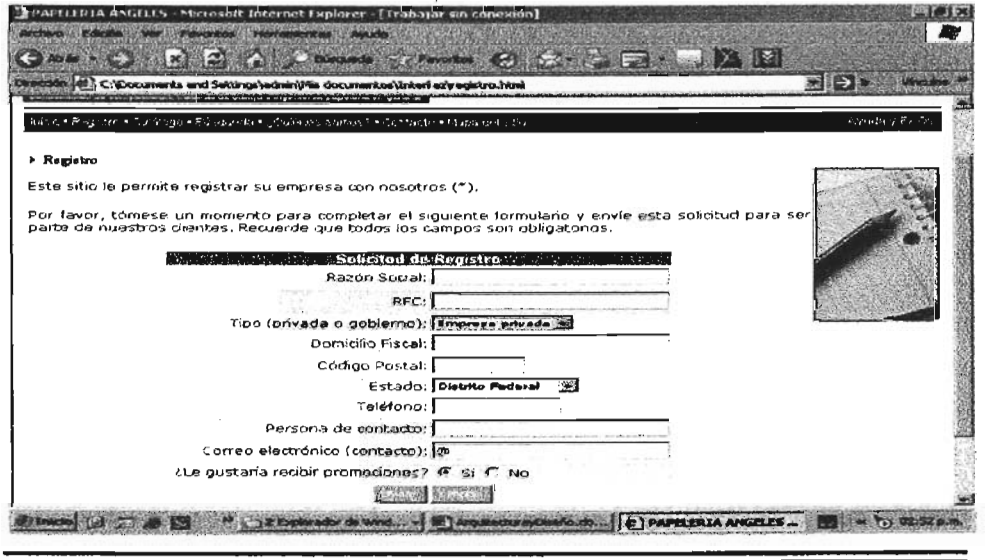
4.6.1.1 Página de Inicio

Desde la página principal existen ligas que permiten al usuario navegar por toda la página, y acceder al catálogo, pero no se permite realizar pedidos, para ello se solicita un nombre y contraseña, después se realiza la validación y si ésta falla, se vuelve a pedir que se introduzca la contraseña, y también se sugiere que se registre en el sistema, en caso de no contar un nombre y contraseña para poder acceder al sistema de pedidos.

Contiene la información básica de la empresa y anuncios de las promociones vigentes que la empresa ofrece.

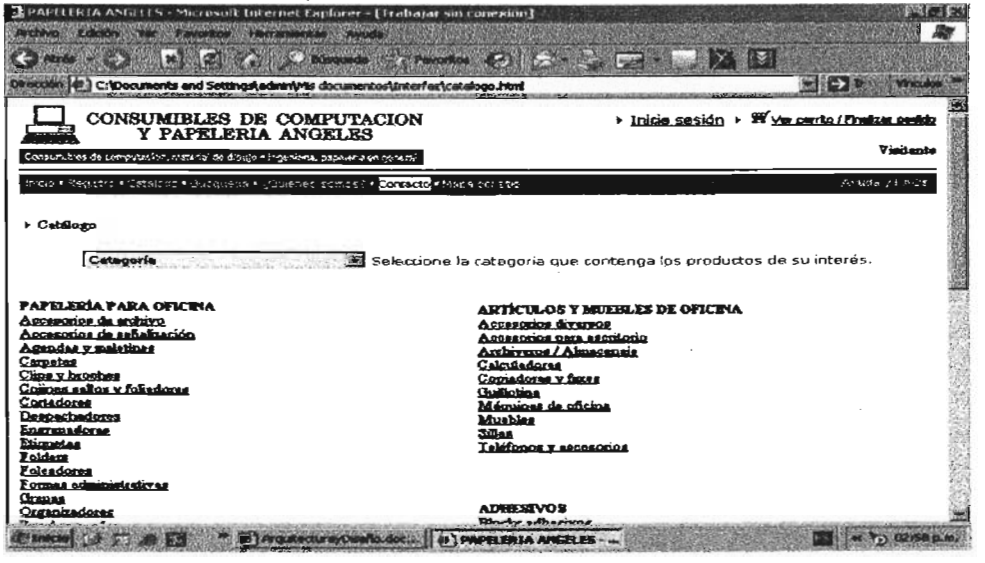
Página de Registro

Ésta página esta formada por un formulario que debe ser llenado por el cliente, para ser registrado en el sistema y así realizar funciones avanzadas, como es levantar pedidos, actualizar datos, etc



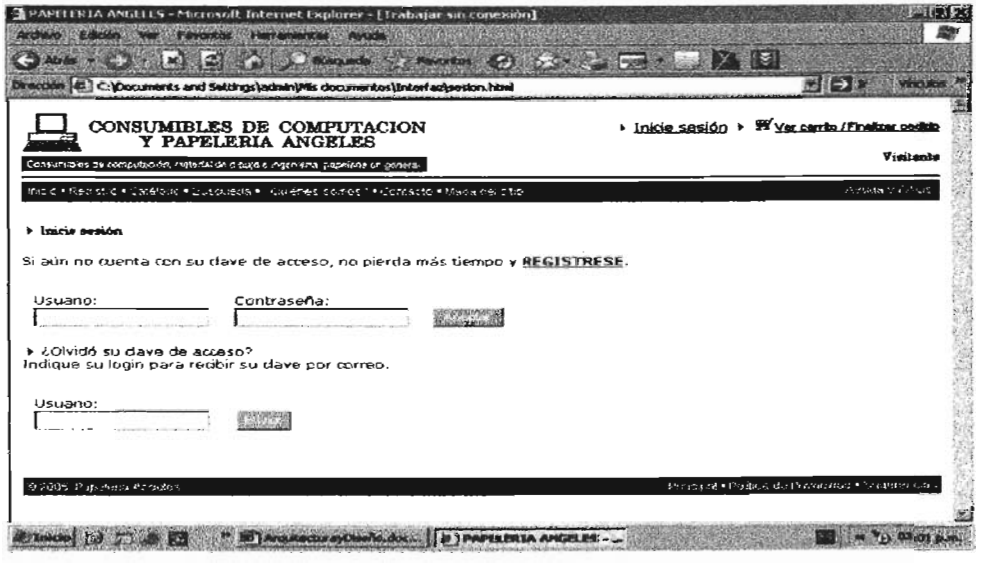
Catálogo de Productos

Éste muestra la lista de categorías de los productos que la empresa ofrece.



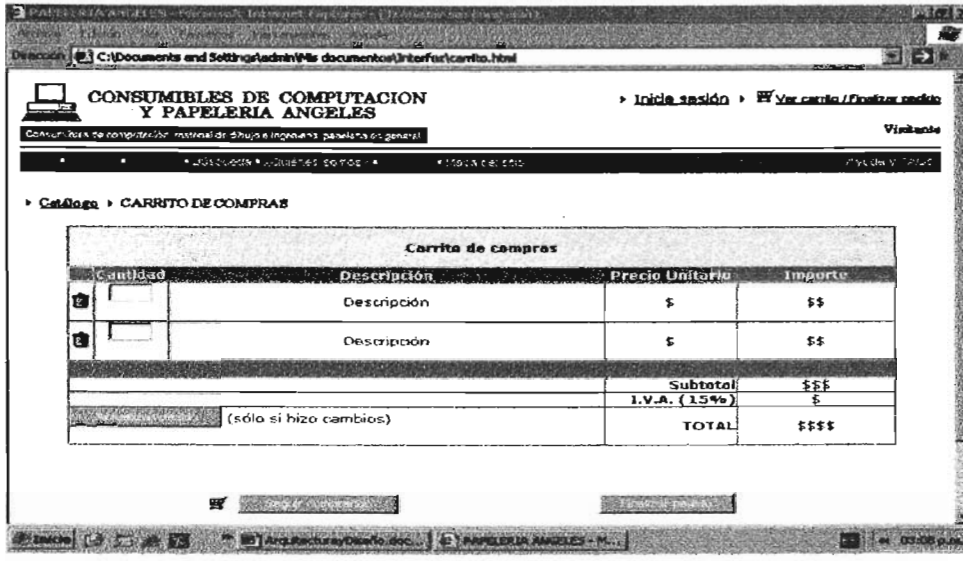
Página de inicio de sesión

En ésta se solicita el nombre de usuario y la contraseña para acceder al sistema, e iniciar una sesión.



Carrito de Compra

Aquí se van acumulando los productos que se eligen del catálogo con la opción comprar.



4.6.2 Codificación en HTML de la Interfaz

Página Principal

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>PAPELERIA ANGELES</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link href="hoja.css" rel="stylesheet" type="text/css">
<STYLE type=text/css>
A:link {COLOR: #ffffff; TEXT-DECORATION: none}
A:active {COLOR: #ffffff; TEXT-DECORATION: none}
A:visited {COLOR: #ffffff; TEXT-DECORATION: none}
A:hover {BACKGROUND: #ffffff; COLOR: #333333; TEXT-DECORATION: none}
</STYLE>
</head>
<body bgcolor="#FFFFFF">
<table width="100%" border="0" cellspacing="0" cellpadding="0">
```

```
 <tr>
  <td><img src="logo.gif" width="336" height="64"></td>
        <td align="right"><p><img src="flecha.gif" width="16" height="11"><span
class="texto"><a href="sesion.html"><font color="#000000"><u>Inicie
     sesi&oacute;n</u></font></a> <img src="flecha.gif" width="16"
height="11"><img src="carr2.gif" width="19" height="14"></span><span
class="fecha"><a href="carrito.html"><font color="#000000"><u>Ver
     carrito / Finalizar pedido</u></font></a></span></p>
    <p class="titulo">Visitante</p></td>
 </tr>
</table>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
 <tr>
  <td height="42" class="link" background="fondo_up.gif">   Inicio &#8226;
   <a href="registro.html">Registro</a> &#8226; <a
href="catalogo.html">Cat&aacute;logo</a>
   &#8226; <a href="busqueda.html">B&uacute;squeda</a> &#8226; <a
href="quienes.html">&iquest;Qui&eacute;nes
   somos?</a> &#8226; <a href="contacto.html">Contacto</a> &#8226; <a
href="mapa.html">Mapa
   del sitio</a>
  <td height="42" class="link" align="right" background="fondo_up.gif"> <a
href="ayuda.html">Ayuda
   y FAQs</a>  </td>
 </tr> <tr>   <td> </td>
  <td align="right" class="fecha"><script language="JavaScript">
   var dayNames = new
Array("Domingo","Lunes","Martes","Miércoles","Jueves","Viernes","Sábado");
   var monthNames = new
Array("Enero","Febrero","Marzo","Abril","Mayo","Junio","Julio","Agosto","Septiembre"
,"Octubre","Noviembre","Diciembre");
   var dt = new Date();
   var y  = dt.getYear();
   // Y2K compliant
   if (y < 1000) y +=1900;
```

```
    document.write(dayNames[dt.getDay()] + ", " + dt.getDate() + " "+
monthNames[dt.getMonth()] + " " + y);
    // -->
     </script></td>
  </tr> <tr>
   <td><img src="flecha.gif" width="16" height="11"><span class="titulo">&iexcl;
    Bienvenidos a nuestro sitio !</span></td>
   <td> </td>
  </tr>
</table>
<p align="justify" class="texto">Somos el <strong>proveedor ideal</strong> para
 su empresa u oficina en el D. F. y &aacute;rea metropolitana. Con nosotros
encontrar&aacute;
 una gran variedad de <strong>art&iacute;culos de oficina</strong>,
 <strong>papeler&iacute;a escolar</strong>, <strong>equipo
 y consumibles de c&oacute;mputo</strong>, todo en <strong>entrega
 inmediata</strong> y hasta la puerta de su negocio <strong>sin
 costo adicional</strong> alguno.</p>
<table width="100%" border="0" align="center">
 <tr>
   <td align="center"><img src="pap_escolar.gif" alt="Papeler&iacute;a escolar"
width="161" height="111"></td>
   <td align="center"><img src="comp_equipo.gif" alt="Equipo de c&oacute;mputo"
width="161" height="111"></td>
   <td align="center"><img src="comp_consum.gif" alt="Consumibles de
c&oacute;mputo" width="161" height="111"></td>
   <td align="center"><img src="pap_ofic.gif" alt="Art&iacute;culos de oficina"
width="161" height="111"></td>
 </tr>
</table>
<p align="justify"> </p>
<p align="justify"><img src="flecha.gif" width="16" height="11"><span
class="titulo">Avisos
 importantes</span> </p>
<p align="justify" class="texto">Despu&eacute;s de ya varios a&ntilde;os en el
```

```
 mercado, ahora contamos con nuestra p&aacute;gina web en la cual, nuestros
clientes  podr&aacute;n navegar, buscar los productos de su preferencia y hacer
pedidos  desde su oficina o negocio.</p>
<p align="justify" class="texto">Y para quienes apenas nos est&aacute;n
conociendo  los invitamos a registrarse para ser parte de nuestro grupo de clientes
y gozar  de los beneficios que les ofrecemos.</p>
<table width="100%" background="fondo_barra.gif" border="0">
 <tr>
  <td class="copy" align="left">&copy; 2005. Papeler&iacute;a Angeles.</td>
  <td class="link" align="right">Principal &#8226; <a
href="suger.html">Sugerencias</a></td>
 </tr>
</table>
</body>  </html>
```

CAPÍTULO 5. REGLAS PARA EL DISEÑO DE LA BASE DE DATOS DEL PROYECTO ORIENTADO A WEB

6.1 Capa de Datos

Es la base de una aplicación de Bases de Datos basada en Web, engloba todos los componentes del sistema que almacenan los datos. Ésta formada básicamente por el Sistema Gestor de Bases de Datos (Data Base Management System) y un paquete encargado de aislar el acceso a los datos.

6.2 Bases de Datos y Modelos de Datos

5.2.1 Modelos de Datos

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones y las restricciones que deben cumplirse sobre los datos. Es decir, un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas; son abstracciones que permiten la implementación de un sistema eficiente de base de datos, además contienen un conjunto de operaciones básicas para la realización de consultas y actualizaciones de datos, e incluyen conceptos para especificar comportamiento.

5.2.2 Bases de Datos

6.2.2.1 Bases de datos relacionales

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. En 1970 fue postulada por Edgar Frank Codd, de los laboratorios IBM en San José (California), su idea fundamental es el uso de "tablas", compuestas de registros (las filas de una tabla) y campos (las columnas de una tabla). En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos. La información puede ser recuperada o almacenada por medio de "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

6.2.2.2 Bases de datos de red

Éste es un modelo, los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos, que son punteros en implementación física . Los registros se organizan como un grafo: los registros son nodos y los arcos son los conjuntos.

6.2.2.3 Bases de datos jerárquicas

Éstas son bases de datos que almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres se le conoce como raíz, y a los nodos que no tienen hijos se les conoce como hojas.

Una de las principales limitaciones de este modelo, es su incapacidad de representar eficientemente la redundancia de datos

6.2.2.4 Bases de datos orientadas a objetos

Éste modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento). Define Una base de datos en términos de objetos, sus propiedades y sus operaciones. Los objetos con la misma estructura y comportamiento pertenecen a una clase, y las clases se organizan en jerarquías. Las operaciones de cada clase se especifican en términos de procedimientos predefinidos llamados métodos.

6.2.2.5 Base de datos distribuidas

Una base de datos distribuida (BDD) es la unión de las bases de datos con redes. La base de datos está almacenada en varias computadoras conectadas en red, (ya sea en el mismo lugar físicamente o distribuidas a lo largo de la red) lo que permite al acceso de datos desde diferentes máquinas. Son la evolución de los cliente-servidor.

La razón principal detrás de las BDD son los organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a la información, sin tener todo centralizado en un solo punto. Ejemplo: bancos, cadenas de hoteles, campus de distintas universidades, sucursales de tiendas departamentales, etc.

5.2.3 Base de Datos

Una Base de datos está definida como un conjunto integrado de datos que modelan un universo dado. Éste universo esta compuesto por objetos interrelacionados. Los objetos de un mismo tipo constituyen una entidad y el lazo que existe entre entidades se denomina asociación. De forma más simple, una base de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior.

5.2.4 Características de los SGBD

1. Separación entre los programas y los datos.
2. El manejo de múltiples vistas por parte de los usuarios
3. El uso de un catálogo para almacenar el esquema de la base de datos.

5.2.5 Funciones de los SGBD

1. Debe permitir la definición de todos los datos.
2. Debe permitir manipular los datos.
3. Debe establecer controles de seguridad de estos datos.
4. Debe permitir accesos concurrentes.

5.2.6 Objetivos de los SGBD

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**. Los usuarios de los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- **Independencia**.
- **Redundancia mínima**. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- **Consistencia**. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad**. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad**. Los SGBD mantienen el conjunto de datos íntegros, es decir evitan que los datos sean modificados o corrompidos.
- **Respaldo y recuperación**. Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos.
- **Control de la concurrencia.** Permiten tener control del número de conexiones que se realizan a la Base de Datos, impidiendo que se modifiquen datos.
- **Tiempo de respuesta**. Un SGBD proporciona un menor tiempo de respuesta de respuesta al realizar consultas a la Base de Datos.

5.2.7 Concepción de una Base de Datos

El ciclo de vida de una base de datos puede descomponerse de 3 etapas:

Concepción:

Consiste en reproducir el mundo real con ayuda de uno de los modelos de datos conocidos, su resultado es un esquema escrito no interpretable por el SGBD.

Creación de la BD vacía:

Se trata de traducir éste esquema en ordenes comprensibles para el SGBD, y se obtiene la estructura de la base de datos sin información.

Explotación:

En ésta fase los registros serán manipulados con la ayuda de los lenguajes de programación. Ahora los usuarios pueden consultar los datos y ponerlos a punto durante el resto de la vida de la base de datos.

6.3 SQL para programadores.

5.3.1 El lenguaje SQL

Es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre Bases de Datos, Tablas y Datos o sobre la estructura de los mismos.
SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática.

5.3.2 Sentencias de selección o consultas.

Las sentencia SELECT recupera datos de una base de datos y los devuelve en forma de resultados de consulta.

5.3.3 Funciones de agrupamiento

Las funciones de agrupamiento devuelven un único valor de un conjunto de registros. Las funciones de agrupamiento permitidas son:

SUM: Devuelve la suma total de los valores de una expresión de columna.
AVG: Devuelve la media de los valores de una expresión de columna.
COUNT: Devuelve el número de valores de una expresión de columna.
MAX Devuelve el valor más alto de los contenidos en una expresión de columna.
MIN Devuelve el valor más bajo de los contenidos en una expresión de columna.

6.3.3.1 Cláusula FROM

Ésta cláusula lista las tablas que contienen los datos a recuperar por la consulta.

6.3.3.2 Cláusula WHERE

Ésta cláusula contiene condiciones que deben de cumplir los registros para ser mostrados en la consulta.

6.3.3.3 Cláusula ORDER BY

Ésta cláusula ordena los resultados de la consulta en base a los datos de una o más columnas.

6.3.3.4 Cláusula GROUP BY

Especifica una consulta sumaria, es decir, en vez de producir una fila de resultados por cada fila de datos de la base de datos, agrupa todas las filas similares y luego

produce una fila sumaria de resultados para cada grupo. Después de GROUP BY se especifican los nombres de uno o más campos cuyos resultados desean sean agrupados.

6.3.3.5 Cláusula HAVING

Ésta cláusula indica que sólo se incluyan ciertos grupos producidos por la cláusula GROUP BY en los resultados de consulta. Utiliza una condición de búsqueda para especificar los grupos deseados, y sólo es válida si previamente se ha especificado la cláusula GROUP BY.

5.3.4 Alias de las Tablas

Los alias son un instrumento para abreviar los nombres de las tablas o fichero y poder referirse a ellos en toda la sentencia.

5.3.5 Creación de una tabla

La sentencia para crear una tabla tiene la forma:
CREATE TABLE nombre_tabla (definición_ columna, ..)
Donde definición_columna está compuesto por el nombre de la columna y el tipo de dato, una tabla puede tener tantas columnas como se deseen.

5.3.6 Destrucción de una tabla

El formato para destruir o borrar una tabla es: DROP TABLE nombretabla.

5.3.7 Sentencia INSERT

Sentencia utilizada para añadir registros a las tablas de la base de datos. El formato de la sentencia es: INSERT INTO nombretabla [{nombrecolumna, ..}] VALUES (valores,–)

5.3.8 Sentencia UPDATE

Sentencia que se utiliza para cambiar o actualizar el contenido de los registros de una tabla de la base de datos. Su formato es: UPDATE nombretabla SET nombrecolumna = valor,...

[WHERE {condición}] es opcional, y determina que registros se modificaran.

5.3.9 Sentencia DELETE

Ésta sentencia se usa para borrar registros de una tabla, su formato es:

DELETE FROM nombretabla [WHERE {condición}]

5.3.10 Llave primaria

Una llave primaria en una tabla está compuesta de uno o más campos consecutivos de la tabla, cada tabla solamente puede tener una llave primaria. La llave primaria identifica a cada registro en la tabla. Para añadir una llave primaria después de que la tabla fue creada es así:
ALTER TABLE tabla ADD constraint llave PRIMARY KEY(llave)

5.3.11 Llave foránea

Una llave foránea es un campo en una tabla que hace referencia a otra tabla. Para añadir una llave foránea a una tabla se hace:
ALTER TABLE tabla ADD constraint llave_Fk FOREIGN KEY(llave_Fk) REFERENCES tabla(llave_Fk)

Nota: el campo debe existir en la tabla que sé referencia sino se necesita crear.

6.4 API JDBC

JDBC es el acrónimo de *Java Database Connectivity*, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar.

El API JDBC hace posible 3 cosas:

- Establecer una conexión con una base de datos o tener acceso a cualquier fuente de datos.
- Ejecutar sentencias SQL
- Procesar los resultados de las consultas

5.4.1 Tipos de Driver JDBC

Tipo 4: Direct-to-Database Pure Java Driver

Éste estilo de driver[1] traduce directamente las llamadas JDBC[2] al protocolo de red utilizado por el DBMS[3], permitiendo una llamada directa de la máquina cliente al servidor DBMS y proporciona una solución básica al acceso por Internet.

Sus ventajas principales son la eliminación de intermediarios y su funcionamiento tanto en applets[4] como en aplicaciones. Sin embargo su principal desventaja es su dificultad para ser implementando por los proveedores de bases de datos.

Tipo 3: Driver para Middleware de Bases de Datos

Traduce las llamadas JDBC a un protocolo de software de tipo middleware[5] y después es traducido al protocolo DBMS. Éste provee conectividad a distintos tipos de bases de datos. Sus ventajas son su flexibilidad para conectarse con diferentes motores de bases de datos y su funcionamiento tanto en aplicaciones como applets. Pero su desventaja principal es el uso de una máquina intermedia, lo cual hace más lenta la transferencia de datos.

5.4.2 Tipo 2: Driver mixto Java-API Nativo

Convierte las llamadas JDBC a las llamadas de tipo nativo DBMS y requiere que algún tipo de código binario sea cargado en la máquina cliente. Su ventaja es su fácil implementación por los proveedores de base de datos, aunque no se puede usar en applets debido al llamado a métodos nativos.

5.4.3 Tipo 1: Puente JDBC-ODBC

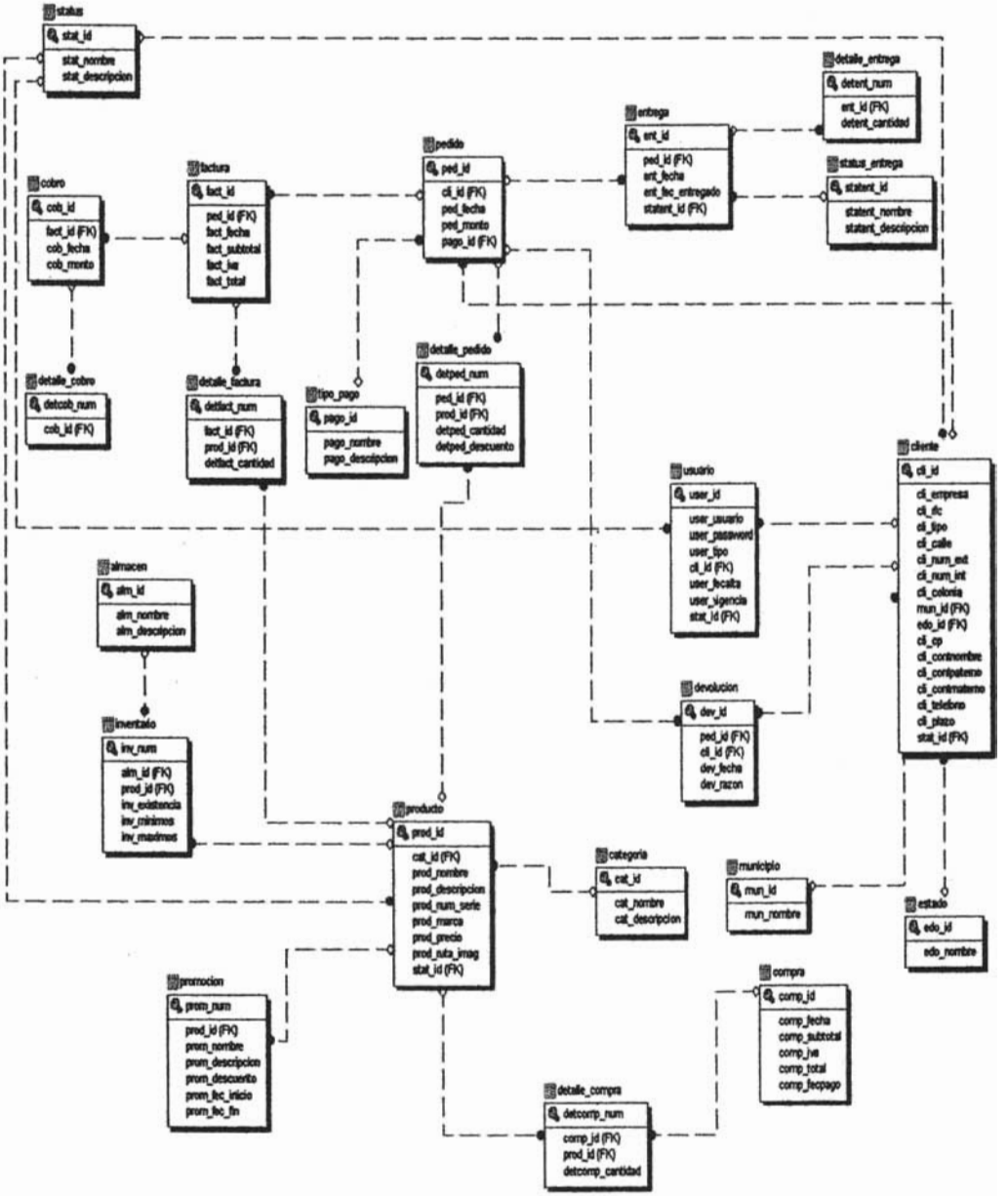
Ésta combinación proporciona el acceso JDBC vía driver ODBC. Su ventaja radica en que la conectividad con Java se puede hacer inmediatamente, sin embargo no funciona en applets, además la conexión se vuelve lenta por el uso de ODBC.

---

[1] Controlador de dispositivos electrónicos.
[2] API JDBC (Java Data Base Conectivity), estándar para la conectividad independiente a bases de datos.
[3] DBMS, Sistemas Manejadores de Bases de Datos
[4] miniaplicaciones Java que se incluyen en las páginas Web y se ejecutan en el navegador Web.
[5] Software de comunicaciones que reside físicamente en el cliente remoto y en un servidor de comunicaciones, localizado entre el cliente y el servidor de aplicaciones. Es el software que actúa como un traductor universal entre distintas tecnologías de radiofrecuencia y protocolos

6.5 Base de Datos del Proyecto

5.5.1 Diagrama Entidad-Relación de la Base de Datos



5.5.2 Diccionario de Datos

Entidad	
Nombre	**Definición**
almacén	Ésta tabla contiene los datos del almacén
categoría	Ésta tabla contiene los datos de la categoría
cliente	Ésta tabla contiene los datos del cliente
cobro	Ésta tabla contiene los datos del cobro sobre facturas
compra	Ésta tabla contiene la información de las compras realizadas
detalle_cobro	Ésta tabla contiene información detallada de los cobros
detalle_compra	Ésta tabla contiene información detallada de las compras realizadas
detalle_entrega	Ésta tabla contiene información detallada de las entregas
detalle_factura	Ésta tabla contiene información detallada de las facturas
detalle_pedido	Ésta tabla contiene la información detallada del pedido
devolucion	Ésta tabla contiene los datos de las devoluciones
entrega	Ésta tabla contiene los datos de las entregas
estado	Ésta tabla es un catálogo que contiene los estados de la república
factura	Ésta tabla almacena los datos de las facturas
inventario	Ésta tabla se almacenan los datos sobre existencia de productos
municipio	Ésta tabla es un catálogo que contiene los municipios del distrito y estado de México
pedido	Ésta tabla contiene los datos del pedido
producto	En esta tabla se almacenan los datos del producto, como son marca, precio, etc.
promocion	Ésta tabla almacena los datos de las promociones realizadas
status	Ésta tabla es un catálogo de status posibles en facturas, pedidos, usuarios, etc.
status_entrega	Ésta tabla almacena la relación entre las entregas y su status dentro del sistema
tipo_pago	Ésta tabla almacena los tipo de pago posibles
usuario	Ésta tabla contiene los datos del la cuenta de usuario del cliente

Atributos de "almacén"			
Nombre	**Definición**	**Es PK**	**Es FK**
alm_id	Éste campo almacena el número de identificación del almacén	SI	No
alm_nombre	Éste campo almacena el nombre del almacén al que nos referimos	No	No
alm_descripción	Éste campo contiene una breve descripción del almacén.	No	No

Atributos de "categoría"			
Nombre	**Definición**	**Es PK**	**Es FK**
cat_id	Éste campo contiene el número de identificación de la categoría	Sí	No
cat_nombre	Éste campo contiene el nombre de la categoría	No	No
cat_descripción	Éste campo contiene una breve descripción de la categoría	No	No

Atributos de "cliente"			
Nombre	**Definición**	**Es PK**	**Es FK**
cli_id	Éste campo contiene el número de identificación del cliente	Sí	No
cli_empresa	Éste campo contiene el nombre de la empresa cliente	No	No
cli_rfc	Éste campo contiene el RFC de la empresa cliente	No	No
cli_tipo	Éste campo contiene el tipo de Empresa cliente del que se trata, que puede ser: Gobierno, Privada	No	No
cli_calle	Éste campo contiene la calle del domicilio de la empresa cliente	No	No
cli_num_ext	Éste campo contiene el número exterior de la calle del domicilio de la empresa cliente	No	No
cli_num_int	Éste campo contiene el número interior de la calle del domicilio de la empresa cliente	No	No
cli_colonia	Éste campo contiene la colonia del domicilio de la empresa cliente	No	No
mun_id	Éste campo almacena el número de municipio	No	Sí
edo_id	Éste campo contiene el número de identificación del estado del domicilio	No	Sí
cli_cp	Éste campo contiene el código postal del domicilio de la empresa cliente	No	No
cli_contnombre	Éste campo contiene el nombre del contacto de la empresa cliente	No	No
cli_contpaterno	Éste campo contiene el apellido paterno de contacto de la empresa cliente	No	No
cli_contmaterno	Éste campo contiene el apellido materno del contacto de la empresa cliente	No	No
cli_telefono	Éste campo contiene el teléfono del contacto de la empresa cliente	No	No
cli_plazo	Éste campo contiene el plazo de pago de la empresa cliente	No	No
stat_id	Éste campo almacena el número de identificación de status	No	Sí

Atributos de "cobro"			
Nombre	**Definición**	**Es PK**	**Es FK**
cob_id	Éste campo contiene el número de identificación del cobro de pedido.	Sí	No
fact_id	Éste campo contiene el número de identificación de la factura.	No	Sí
cob_fecha	Éste campo contiene la fecha en que se realizara el cobro del pedido	No	No
cob_monto	Éste campo contiene el monto del cobro de pedido	No	No

Atributos de "compra"			
Nombre	**Definición**	**Es PK**	**Es FK**
comp_id	Éste campo contiene el número de identificación de la compra	Sí	No
comp_fecha	Éste campo contiene la fecha en que se realizo la compra	No	No
comp_subtotal	Éste campo contiene el subtotal de la compra, es decir el total sin IVA	No	No
comp_iva	Éste campo contiene el IVA de la compra	No	No
comp_total	Éste campo contiene el monto total de la compra, con IVA incluido	No	No
comp_fecpago	Éste campo contiene la fecha de pago de la compra.	No	No

Atributos de "detalle_cobro"			
Nombre	**Definición**	**Es PK**	**Es FK**
detcob_num	Éste campo contiene el número de identificación del detalle de cobro	Sí	No
cob_id	Éste campo contiene el número de identificación del cobro de pedido	No	Sí

Atributos de "detalle_compra"			
Nombre	**Definición**	**Es PK**	**Es FK**
detcomp_num	Éste campo contiene en número de identificación del detalle de la compra	Sí	No
comp_id	Éste campo contiene el número de identificación de la compra	No	Sí
prod_id	Éste campo contiene el número de identificación del producto. El cual consta de 6 dígitos los primeros 2 dígitos corresponden a la categoría y subcategoría y el resto al número de producto	No	Sí

Atributos de "detalle_compra"			
Nombre	**Definición**	**Es PK**	**Es FK**
detcomp_cantidad	Éste campo contiene la cantidad de productos que se compraron	No	No

Atributos de "detalle_entrega"			
Nombre	**Definición**	**Es PK**	**Es FK**
detent_num	Éste campo contiene el número de detalle de entrega	Sí	No
ent_id	Éste campo contiene el número de identificación de la entrega	No	Sí
detent_cantidad	Éste campo contiene la cantidad de productos entregados	No	No

Atributos de "detalle_factura"			
Nombre	**Definición**	**Es PK**	**Es FK**
detfact_num	Éste campo contiene el número de detalle de factura	Sí	No
fact_id	Éste campo almacena el número de identificación de la factura	No	Sí
prod_id	Éste campo contiene el número de identificación del producto. El cual consta de 6 dígitos los primeros 2 dígitos corresponden a la categoría y subcategoría y el resto al número de producto	No	Sí
detfact_cantidad	Éste campo contiene la cantidad de productos de la factura	No	No

Atributos de "detalle_pedido"			
Nombre	**Definición**	**Es PK**	**Es FK**
detped_num	Éste campo contiene el número de identificación del detalle del pedido	Sí	No
ped_id	Éste campo almacena el número de pedido	No	Sí
prod_id	Éste campo contiene el número de identificación del producto. El cual consta de 6 dígitos los primeros 2 dígitos corresponden a la categoría y subcategoría y el resto al número de producto	No	Sí
detped_cantidad	Éste campo contiene la cantidad de productos del pedido	No	No
detped_descuento	Éste campo contiene el descuento de los productos, puede ser nulo, en caso de no haber descuento.	No	No

Atributos de "devolucion"			
Nombre	**Definición**	**Es PK**	**Es FK**
dev_id	Éste campo contiene el número de identificación de la devolución	Sí	No
ped_id	Éste campo contiene el número de pedido	No	Sí
cli_id	Éste campo contiene el número de identificación del cliente	No	Sí
dev_fecha	Éste campo contiene la fecha en que se realizo la devolución del producto	No	No
dev_razon	Éste campo contiene la razón por la cual se devolvió el producto.	No	No

Atributos de "entrega"			
Nombre	**Definición**	**Es PK**	**Es FK**
ent_id	Éste campo contiene el número de identificación de la entrega	Sí	No
ped_id	Éste campo almacena el número de pedido	No	Sí
ent_fecha	Éste campo contiene la fecha limite en que se entregará el pedido	No	No
ent_fec_entregado	Éste campo contiene la fecha real en que se entrego el pedido	No	No
statent_id	Éste campo almacena el número de status de entrega	No	Sí

Atributos de "estado"			
Nombre	**Definición**	**Es PK**	**Es FK**
edo_id	Éste campo contiene el número de identificación del estado del domicilio	Sí	No
edo_nombre	Éste campo contiene el nombre del estado al que se refiere	No	No

Atributos de "factura"			
Nombre	**Definición**	**Es PK**	**Es FK**
fact_id	Éste campo almacena el número de identificación de la factura	Sí	No
ped_id	Éste campo almacena el número de pedido	No	Sí
fact_fecha	Éste campo contiene la fecha en que se realizo la factura	No	No
fact_subtotal	Éste campo contiene el subtotal de la factura	No	No
fact_iva	Éste campo contiene el IVA de la factura	No	No
fact_total	Éste campo contiene el Total de la factura	No	No

Atributos de "inventario"			
Nombre	**Definición**	**Es PK**	**Es FK**
inv_num	Éste campo contiene el número de inventario	Sí	No
alm_id	Éste campo almacena el número de identificación del almacén	No	Sí
prod_id	Éste campo contiene el número de identificación del producto. El cual consta de 6 dígitos los primeros 2 dígitos corresponden a la categoría y subcategoría y el resto al número de producto	No	Sí
inv_existencia	Éste campo contiene las existencias reales en almacén, cuantos productos hay en almacén.	No	No
inv_minimos	Éste campo contiene el número mínimo de productos que pueden haber en el almacén	No	No
inv_maximos	Éste campo contiene el número máximo de productos que puede haber en almacén	No	No

Atributos de "municipio"			
Nombre	**Definición**	**Es PK**	**Es FK**
mun_id	Éste campo almacena el número de municipio	Sí	No
mun_nombre	Éste campo almacena el nombre del municipio	No	No

Atributos de "pedido"			
Nombre	**Definición**	**Es PK**	**Es FK**
ped_id	Éste campo almacena el número de pedido	Sí	No
cli_id	Éste campo contiene el número de identificación del cliente	No	Sí
ped_fecha	Éste campo contiene la fecha en que se realizo el pedido	No	No
ped_monto	Éste campo contiene el monto total del pedido	No	No
pago_id	Éste campo almacena el número de tipo de pago	No	Sí

Atributos de "producto"			
Nombre	**Definición**	**Es PK**	**Es FK**
prod_id	Éste campo contiene el número de identificación del producto. El cual consta de 6 dígitos los primeros 2 dígitos corresponden a la categoría y subcategoría y el resto al número de producto.	Sí	No
cat_id	Éste campo contiene el número de identificación de la categoría	No	Sí
prod_nombre	Éste campo contiene el nombre del producto o producto.	No	No
prod_descripción	Éste campo contiene una breve descripción del producto o producto	No	No
prod_num_serie	Éste campo contiene el número de serie del	No	No

Atributos de "producto"			
Nombre	**Definición**	**Es PK**	**Es FK**
	producto.		
prod_marca	Éste campo contiene la marca del producto.	No	No
prod_precio	Éste campo contiene el precio del producto.	No	No
prod_ruta_imag	Éste campo contiene la ruta de la imagen del producto.	No	No
stat_id	Éste campo almacena el número de status	No	Sí

Atributos de "promocion"			
Nombre	**Definición**	**Es PK**	**Es FK**
prom_num	Éste campo contiene el número de identificación de la promoción	Sí	No
prod_id	Éste campo contiene el número de identificación del producto. El cual consta de 6 dígitos los primeros 2 dígitos corresponden a la categoría y subcategoría y el resto al número de producto	No	Sí
prom_nombre	Éste campo contiene el nombre de la promoción	No	No
prom_descripción	Éste campo contiene la descripción de la promoción	No	No
prom_descuento	Éste campo contiene el descuento de la promoción para el producto o productos.	No	No
prom_fec_inicio	Éste campo contiene la fecha de inicio de la promoción	No	No
prom_fec_fin	Éste campo contiene la fecha de termino de la promoción	No	No

Atributos de "status"			
Nombre	**Definición**	**Es PK**	**Es FK**
stat_id	Éste campo almacena el número de identificación de status	Sí	No
stat_nombre	Éste campo almacena el nombre del status	No	No
stat_descripción	Éste campo almacena una breve descripción del status	No	No

Atributos de "status_entrega"			
Nombre	**Definición**	**Es PK**	**Es FK**
statent_id	Éste campo almacena el número de status de entrega	Sí	No
statent_nombre	Éste campo almacena el nombre del status	No	No
statent_descripción	Éste campo almacena una breve descripción del status	No	No

Atributos de "status_entrega"			
Nombre	**Definición**	**Es PK**	**Es FK**
Atributos de "tipo_pago"			
Nombre	**Definición**	**Es PK**	**Es FK**
pago_id	Éste campo almacena el número de tipo de pago	Sí	No
pago_nombre	Éste campo almacena el nombre del tipo de pago	No	No
pago_descripción	Éste campo almacena una breve descripción del tipo de pago.	No	No

Atributos de "usuario"			
Nombre	**Definición**	**Es PK**	**Es FK**
user_id	Éste campo almacena el número de identificación de usuario	Sí	No
user_usuario	Éste campo almacena el nombre de usuario del sistema	No	No
user_password	Éste campo almacena la clave del usuario	No	No
user_tipo	Éste campo almacena el tipo de usuario que se trata; puede ser cliente, cliente avanzado o administrador	No	No
cli_id	Éste campo contiene el número de identificación del cliente	No	Sí
user_fecalta	Éste campo contiene la fecha de alta en el sistema.	No	No
user_vigencia	Éste campo almacena la fecha de vigencia del usuario	No	No
stat_id	Éste campo almacena el número de identificación de status	No	Sí

5.1.1 Especificaciones de la Base de Datos del Sistema

La Base de Datos del sistema fue creada en MySQL versión 4.1.0. El control de la base de datos se realizará mediante un login, es decir, un nombre y una contraseña de usuario, que servirá para realizar la autentificación en el sistema y permitirá acceder a éste.

El nombre y la contraseña de usuario tendrá de 3 niveles:

- General (público en general), acceso a catálogo de productos e información general.
- Intermedio (Usuarios Registrados), acceso a catálogo de productos, información general y al sistema de pedidos.
- Total (Administradores), acceso total a todo el sistema.

CAPÍTULO 6. APLICACIÓN DE LAS REGLAS DEL NEGOCIO CON STRUTS E IBATIS.

6.1 REGLAS DE NEGOCIO

6.1.1 Definición

Las reglas de Negocio son las reglas que debe seguir una aplicación para que refleje las restricciones que existen en el negocio dado, de modo que nunca sea posible llevar a cabo acciones no válidas, como crear facturas a clientes inexistentes.

Éstas reglas garantizan que la aplicación refleje parte el funcionamiento real del negocio, para automatizar tareas y llevarlas a cabo de modo eficiente. Además de que su aplicación permite controlar el acceso y manipulación de los datos, para asegurar que la información suministrada por la aplicación sea correcta.

6.1.2 Clasificación de reglas de negocio

Éstas se clasifican en varios grupos, pero los principales son:

5.1 Reglas de Modelo de datos.

Éste grupo se compone de todas aquellas reglas que se encargan de controlar la información almacenada para cada atributo o propiedad de una entidad u objeto, éstas especifican los valores permitidos para cada atributo de las diversas entidades que se almacenan, es decir que indican los valores que acepta cada campo de cada tabla de la Base de Datos de sistema.

Por ejemplo, no hay precios de productos negativos, el sexo de una persona solo puede ser femenino o masculino, las fechas deben fechas validas, etc.

5.2 Reglas de Relación

Éste grupo incluye todas aquellas reglas que controlan las relaciones entre los datos, estas especifican que relaciones existen entre que objetos y como se lleva a cabo esta relación. Por ejemplo, un pedido debe ser realizado por un cliente existente y que esté registrado en el sistema, no pueden realizarse facturas a clientes inexistentes, un cliente no puede ser eliminado si tiene pedidos pendientes, etc.

5.3 Reglas de Derivación

Éste grupo contiene el conjunto de reglas que especifican y controlan la obtención de información que se puede calcular a partir de otra ya existente. Por ejemplo, el total de un pedido se calcula a partir de diversos campos como son el subtotal, el IVA, el número de unidades vendidas, el precio por unidad, etc.

5.4 Reglas de Restricción

Éste grupo lo conforman aquellas reglas que restringen los datos que el sistema puede contener, éstas son parecidas a las reglas del modelo de datos, puesto que también impiden la introducción de datos erróneos, pero con la diferencia que éstas restringen el valor de los atributos o propiedades de cada entidad, para lo cual requieren para su verificación del acceso a otros fragmentos de información.

5.5 Reglas de Flujo

Éste último grupo incluye todas aquellas reglas que determinan y limitan como fluye la información a través de un sistema, es decir indican qué camino recorre la información y obligan a que se sigan sólo caminos válidos.

6.1.3 Donde ubicar las reglas de negocio

La ubicación de las reglas depende del grupo al que la regla pertenezca, ya que su tipo indica el lugar en que se deben ubicar en la arquitectura de 3 Capas.

Reglas de modelo de datos: Deben ubicarse en el servidor, ya que especifican los valores válidos de cada atributo(campo) de las diferentes entidades(tablas) que se almacenan en la Base de Datos, esto se logra eligiendo correctamente el tipo de campos de cada tabla de acuerdo al tipo de datos que almacenan. Aunque también deben implementarse validaciones a nivel cliente, esto se hace validando los datos que se introducen en la forma HTML, para evitar almacenar datos erróneos, producir mensajes de error que alerten al cliente y así pueda corregir los datos antes de ser enviados

Reglas de relación: Deben ser ubicadas en el servidor, pues se pueden implementar fácilmente, ya que los gestores de base de datos proporcionan integridad referencial y mecanismos necesarios y las validaciones se hacen de forma muy rápida.

Reglas de derivación: Éstas al ser mayormente cálculos, lo ideal sería implementarlas en el servidor, para que la información que viaje hasta el cliente sea solamente el resultado, pero algunas reglas son muy complejas, por tanto se pueden implementar en la capa intermedia, en una aplicación.

Reglas de restricción: Deben implementarse en el servidor o en la capa intermedia, porqué consideran restricciones en los datos que dependen de información presente en varias tablas, por lo que su implementación debe ser la más cercana a los datos, pero como el lenguaje SQL posee limitaciones, la mejor opción sería implementarlas en la capa intermedia.

Reglas de flujo: Éstas son implementadas generalmente en la capa intermedia, pues suelen ser bastante complejas para ser manejadas por un gestor de bases de datos.

6.2 Struts

6.2.1 ¿Qué es Struts?

Es un *framework* de desarrollo favorecido por la Fundación Apache de Software,[1] forma parte del proyecto Jakarta[2], su primera versión fue desarrollada entre mayo 2000 y junio del 2001. Struts[3] es una solución para manejar Aplicaciones Web.

Un *framework* es una serie de interfaces que son diseñadas para trabajar en conjunto y manejar un tipo particular de problema, su objetivo es ayudar a los programadores en el desarrollo y mantenimiento de aplicaciones complejas.

6.2.2 Componentes principales de Struts

Struts cuenta con componentes que controlan la ejecución y validan la entrada de datos y estos son:

- ActionServlet: Controla el flujo de ejecución, es el controlador.
- Action: Se utiliza para acceder a los objetos de negocio, pues cuando el ActionServlet recibe una petición usa el URI para determinar la acción que usará para resolver dicha petición.

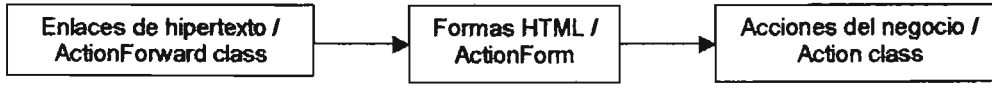
[1] *Apache Software Foundation*
[2] http://jakarta.apache.org/
[3] Struts es un API de Java, que funciona como una herramienta de soporte para el desarrollo de aplicaciones Web.

- ActionForm: Valida la entrada de datos, pues el objeto Action necesita conocer los valores enviados en la petición, así que el ActionServlet entrega la entrada en un JavaBean de tipo ActionForm.
- ActionForward: Cada objeto HttpRequest debe ser contestado con un objeto HttpResponse, la redireccion se realiza mediante un objeto tipo ActionForward.
- Struts-config.ximl: Contiene la configuración de la aplicación, incluida la seguridad, el acceso a la base de datos, el mapeo de servlets.

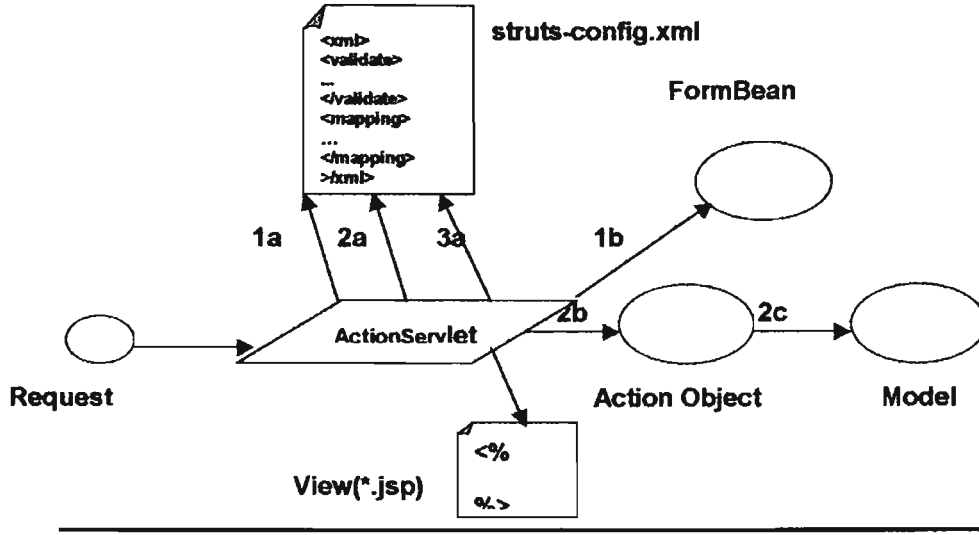
6.2.3 Ventajas de utilizar Struts

a) Enlaces de Hipertexto. Podemos definir un enlace de hipertexto a través de un ActionForward, lo que permite definir el enlace hacia la ruta que debe dirigirse y darle un nombre para usarlo posteriormente, esto proporciona un adecuado control de flujo de las acciones del sistema.

b) Formas de HTML. Struts proporciona una clase tipo ActionForm que sirve para manipular la entrada de una forma HTML, validarla y desplegarla nuevamente al usuario para correcciones junto con los mensajes correspondientes solicitando corregir o completar algún dato, esto facilita el uso de formularios.



6.2.4 Operación del Framework de Struts en una aplicación web



Comienza con el Request de parte del usuario.

1a. El ActionServlet busca en la configuración un ActionForm que corresponda a la solicitud realizada por el usuario mediante el Action marcado en una forma HTML.
1b. El ActionForm seleccionado permite validar la entrada del usuario, además permite instanciar un JavaBean que contiene los datos capturado en la forma HTML para utilizarlos posteriormente.
2a. El ActionServlet busca en la configuración el objeto Action asociado al ActionForm elegido.
2b. Le pasa al Objeto Action, el objeto Request, Response y el ActionForm con los datos capturados y ya validados.
2c. El objeto Action resuelve la petición contenida en el Request, invoca los objetos del negocio que sean requeridos de parte del Modelo.
3a. El ActionServlet localiza en la configuración la vista que debe presentar los resultados de la petición de usuario.
3b. Se invoca la vista definida en la configuración para presentar los resultados de la petición del usuario.

6.3 Ibatis

6.3.1 ¿Qué es Ibatis?

Es una librería ORM (Object Relational Mapping), donde se definen las entidades de la base de datos relacional en un archivo de configuración con formato XML de forma que sea independiente del manejador de base de datos con el que estemos trabajando.

Sirve para realizar el acceso a los datos a través de servicios abstractos, de forma que sean independientes de alguna versión de manejador de base de datos.

La abstracción consiste en definir las rutinas de acceso primarias a la base de datos

- Conectar / Desconectar
- Ejecutar Consultas
- Ejecutar Sentencias DML
  - Insertar
  - Eliminar
  - Actualizar

6.3.2 Procedimiento de Utilización de Ibatis

1. Definir los objetos dentro de la base de datos relacional.

   Create table Persona (numero String(5) , nombre string(30))}
2. Definir los objetos como JavaBeans.

```
package com.orc.ibatis;
public class Persona{
    private string nombre;
    private string numero;
public void setNumero(String numero){
    this.numero= numero;}
public void setNombre(String nombre){
    this.nombre = nombre;}   }
```

3. Definir el acceso a los objetos como XML.
4. Incluir la definición de cada objeto Relacional dentro de la configuración IBATIS, en el archivo sql-map-config.xml.
   - El archivo database.properties indica los parámetros de conexión a la base de datos desde IBATIS.
   - El elemento <sqlMap resource>, se aplica para cada objeto de la base de datos que deseemos manipular desde IBATIS.
5. Definir los parámetros de conexión a la base de datos.

   driver= nombre del driver Java
   username= nombre de usuario
   password= clave
6. Crear la aplicación que hace uso de la librería IBATIS
   - Archivos necesarios para la ejecución de la aplicación, incluyendo la librería IBATIS.
     - ibatis-common-2.jar
     - ibatis-sqlmap-2.jar
     - commons-loggin.jar
     - dom.jar
     - sax.jar
     - xercesImpl-2-4-0.jar
     - xmlParserAPIs-2-4-0.jar

   Para uso de MySQL

- mysql-connector-java-3.0.15-ga-bin.jar

```
public class prueba {
// configuración del mapeo de la base de datos
private statisc SqlMapClient sqlMap = null;
public static final String IBATIS_XML_CONFIG_FILE= "sql-map-config.xml";
static {
    try {   String resource = IBATIS_XML_CONFIG_FILE;
            Reader reader = Resources.getResourceAs Reader(resource);
            catch (Exception e) {
            e.printStackTrace(); }
}
public static SqlMapClient getSqlMapInstance() {
            return sqlMap;  }
public static void main (String[] args) {
try         {                   Persona     alguien=     (Persona)
sqlMap.queryForObject("getPersona","P0001");

System.out.println {
    "Busqueda simple – Persona numero " + alguien.getPersona() +
    "Nombre:  "  +  emp.getNumero()  +  "Numero  persona  :  "  +
alguien.getNumero());
    List list = (List) sqlMap.queryForList("getPersona", "");
    System.out.println("Numero de registros = " + list.size());
    for( Interator iter= list.interator () ; iter.hasNext(); ) {
            Persona element = (Persona) iter.next();
            System.out.println ("Numero persona : " + element.getNumero() +
                "Nombre  : " + element.getPersona());  }
     } catch (Exception e) {    e.printStackTrace(); }
 } //fin main
}//fin clase prueba
```

CONCLUSIÓN

En este trabajo se trataron los temas necesarios para el desarrollo de un proyecto orientado a Web.

En el primer capítulo conocimos los conceptos básicos fundamentales para el desarrollo de un proyecto orientado a Web, repasando el concepto de Internet y la diferencia entre Internet y Web, además de las herramientas necesarias para el desarrollo de un proyecto Web, que son Tomcat, como servidor de aplicación y MySQL como gestor de bases de datos.

En el segundo capítulo se trataron los conceptos el análisis y diseño del proyecto, utilizando herramientas UML, en este caso aprendimos a utilizar la herramienta Poseidón para construir diagramas UML y la manera de realizar casos de uso correctos. La fase de análisis de un proyecto es muy importante y no se puede omitir, ya que sin el análisis adecuado no es imposible construir una aplicación eficiente, porqué no se conocen los requerimientos del sistema, ni la forma en que el sistema debe hacer las tareas para lo cual es diseñado, por ello su importancia.

Después, en el tercer capítulo se conocieron las características y la sintaxis del Lenguaje Java, así como sus tipos de aplicaciones, para posteriormente desarrollar nuestra aplicación. Descubrimos la importancia de Java y su facilidad para programar, ya que cuenta con numerosos módulos o clases que facilitan la orientación a Web, haciendo más eficaz a la aplicación y facilitando su mantenimiento.

Más tarde, en el cuarto capítulo se presentaron las recomendaciones para el diseño de la interfaz de usuario, es decir, la parte del sistema que permitirá el acceso a la aplicación. Se utilizó Dreamweaver como editor HTML, para la creación de la interfaz, y conocimos las tags (etiquetas) más comunes utilizadas para construir páginas HTML y descubrimos la posibilidad de utilizar JavaScript para realizar validaciones y operaciones sencillas en nuestra página.

La interfaz de usuario es la parte más importante en un sistema Web, porqué es la que interactúa con el cliente, lo que él ve, y es muy importante que tenga colores agradables, no agresivos, y sea fácil navegar por ella, con botones o menús de ayuda que lo guíen, y sobre todo que presente la información necesaria en forma clara.

Luego, en el quinto capítulo se vieron las reglas para diseñar y construir la base de datos del proyecto, utilizando el manejador de bases de datos MySQL, y el lenguaje SQL para la creación de las tablas y las consultas, aprendimos a utilizar las sentencias básicas SQL para crear tablas y realizar búsquedas en la base de datos.

La Base de Datos es la parte que almacena la información del sistema, por lo tanto es importante diseñarla correctamente. Un mal diseño de ésta produce inconsistencia en los datos y redundancia, haciendo las consultas ineficientes e incluso mostrando información errónea, para ello se requiere de un buen diseño siguiendo sencillas reglas para evitar estos problemas.

Por último, en el sexto capítulo se vio la aplicación de las reglas del negocio, utilizando los API´s Struts e Ibatis de Java, los cuales permiten la implementación de las reglas del negocio en nuestra aplicación y dan a su vez integridad y seguridad a los datos que se almacenarán en nuestra base de datos, aprendimos su funcionamiento y las ventajas de su utilización en nuestra aplicación.

La aplicación de las reglas del negocio es importante, porque esto permite que la aplicación represente más fielmente las operaciones del mundo real, es decir que como se hacen en realidad, impidiendo acciones ilegales, como saldos y precios negativos, límites de crédito infinitos y cosas así.

EL API Struts es una forma adecuada de implementar reglas del negocio, pues admite validaciones más complejas sobre los datos, lo que con una aplicación simple es muy difícil implementar. Además ofrece seguridad porque separa los datos de la aplicación.

Ibatis es un conjunto de aplicaciones que permite separar la aplicación de la base de datos, es decir, se puede utilizar una misma aplicación con diferentes tipos de bases de datos, ya sean MySQL, Oracle, DB2, etc., no importa porque utilizando Ibatis sólo se modifica una pequeña línea de código para cambiar la base de datos, permitiendo que nuestra aplicación sea transportable y la migración de una base de datos a otra no sea problema, ya que sólo se coloca el controlador adecuado de Java y se indica a que base de datos conectarse y con que usuario y contraseña y listo.

Por tanto al utilizar Ibatis y Struts, se separarán los datos de la aplicación y de la base de datos, proporcionando seguridad, integridad, y aplicando las reglas del negocio necesarias.

Bibliografía consultada

- **Ing. Víctor R. Aguilar**, Apuntes del "2° *Diplomado de Desarrollo de Sistemas en Web* ", Periodo 2004-2005.
- **ARNOLD, Ken y GOSLING, James. Addison-Wesley/Domo.** "*El lenguaje de Programación Java*". Wesley Iberoamericana. 1997. 334 páginas. (*Muy básico, escrito por el desarrollador del lenguaje*).
- **BISHOP**, "*Java - Fundamentos de programación*", Addison-Wesley, 1999
- **BOOCH, G.,** *"El Lenguaje Unificado de Modelado",* Addison Wesley Iberoamericana, 1999
- **CHUCK MUSCIANO, Bill Kennedy**, "*HTML La Guía Completa*", Ed. McGrawHill, 1999
- **CRAING Larman**, "*UML y Patrones, Introducción al Análisis y Diseño Orientado a Objetos*", Prentice-Hall, 1999.
- **DATE, C.J.** "*Introducción a los Sistemas de Bases de Datos.*" Addison-Wesley, 1995.
- **DEITEL**, "*Cómo programar en Java*", Prentice-Hall, 1998
- **FROUFE, Agustín.** "*Java 2. Manual de usuario y tutorial*", Ed. Ra-Ma
- **NAUGHTON, Patrick**. *"Manual de Java"*. Mc. Graw Hill 1996. 395 páginas. (*Introduce todos los aspectos de la programación básica en Java*).
- **RAYA CABRERA, José Luis**, "*HTML 4: Guía de referencia y Tutorial*". Ed. Alfaomega, México D. F. 1999

Referencias en Internet

- Enciclopedia virtual on-line, información general: http://es.wikipedia.org/
- Información General: http://www.lafacu.com/
- Manual básico de Struts: http://www.programacion.net/java/tutorial/joa_struts/
- Página oficial de Struts: http://struts.apache.org/
- Página oficial de *Sun Microsystems* sobre Java: http://java.sun.com/
- Página Web Oficial de SGML: http://www.sgml.org/
- Sitio Oficial de Ibatis: http://ibatis.apache.org/ -
- Sitio oficial de JakartaTomcat: http://jakarta.apache.org/builds/jakarta-tomcat-4.0/
- Sitio oficial de MySQL: http://www.mysql.com/
- Tutorial básico de MySQL: http://www.abcdatos.com/tutoriales/tutorial/l5491
- Tutorial de HTML: http://entren.dgsca.unam.mx/Html/
- Tutorial de MySQL: http://www.fuentelibre.com/
- Tutorial Ibatis: http://www.programacion.com/java/tutorial/hibernate/ -
- Tutorial JakartaTomcat: http://manuales.dgsca.unam.mx/jsp/intro.html
- Tutorial Tomcat: http://www.programacion.com/java/tutorial/tomcatintro/
- Tutoriales y artículos sobre J2ME: http://www.palowireless.com/java/tutorials.asp
- Tutorial Java Nivel Básico: http://www.programacion.com/tutorial/java_basico/