SISTEMA WEB PARA EL MANEJO DE COTIZACIONES, PEDIDOS Y OFERTAS PARA LA EMPRESA PC WARE LTDA

SANDRA MILENA MALDONADO MANRIQUE
JEOVANY ENRIQUE LOPEZ POLO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2005

SISTEMA WEB PARA EL MANEJO DE COTIZACIONES, PEDIDOS Y OFERTAS PARA LA EMPRESA PC WARE LTDA

SANDRA MILENA MALDONADO MANRIQUE
JEOVANY ENRIQUE LOPEZ POLO

Proyecto de grado presentado como requisito parcial para optar al título de Ingeniero de Sistemas

Director
HECTOR NIÑO QUIÑONEZ
Ingeniero de Sistemas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2005

AGRADECIMIENTOS

A DIOS, por su amparo, guía, compañía y fortaleza recibida a través de toda mi carrera.

A mis padres Esperanza Manrique Y Rubén Maldonado, por su amor incondicional, esfuerzo y sacrificio diario, brindándome la oportunidad de estudiar, para así, llegar a convertirme en profesional.

Al Ingeniero Elbert Jaimes Cáceres, representante legal de la empresa PC WARE, por su apoyo y colaboración al permitirnos desarrollar el proyecto de grado en su empresa.

Al Profesor Héctor Niño Quiñonez, director del proyecto, por su apoyo y guía en la elaboración y culminación del presente proyecto.

En memoria del Ingeniero Gilberto Rivas, guía y amigo irremplazable, quien me apoyó en todo momento, me impulsó a seguir adelante y me enseñó a amar mi carrera.

Al Ingeniero Brezhnev Joya, por apoyarme en todo momento, comprenderme e impulsarme a lograr mis objetivos, y por ser esa persona incondicional, que siempre llevo en el corazón.

Y a todas aquellas personas que de una u otra forma colaboraron en el desarrollo del presente proyecto.

AGREDECIMIENTOS

En memoria de mi madre Edilsa Polo y mi amigo Gilberto Rivas que en paz descansen, quienes me enseñaron el verdadero valor de la vida y que siempre hay que seguir delante para alcanzar las metas propuestas.

A Dios por ser mi apoyo, compañía y fortaleza diaria.

A mis abuelos Elvira Meza y José Manuel López por brindarme su amor y apoyo incondicional, sin el cual terminar mi carrera no habría sido posible

A mi padre Manuel López y hermanos Manuel José López, Nelsy López y ShirLey Polo, por su cariño y comprensión.

A Mayra Guerra por su cariño siempre constante, apoyo, y por ser esa persona incondicional que siempre llevo en mi corazón.

Al Ingeniero Elbert Jaimes Cáceres, por brindarnos su apoyo y confianza al permitirnos realizar el proyecto de grado en su empresa.

Al profesor Héctor Niño Quiñónez, por su colaboración en el desarrollo y ejecución del presente proyecto.

Y a todas aquellas personas que de una u otra forma colaboraron en el desarrollo del presente proyecto.

RESUMEN

TITULO: SISTEMA WEB PARA EL MANEJO DE COTIZACIONES, PEDIDOS Y OFERTAS PARA LA EMPRESA PC WARE LTDA*

AUTORES: MALDONADO MANRIQUE, Sandra, y, LOPEZ POLO, Jeovany**

PALABRAS CLAVES: Sitio web, Sistema de Información, aplicaciones cliente servidor, JSP, Struts, bases de datos relacionales, Java, tienda virtual, e-comerce.

DESCRIPCIÓN:

PC WARE LTDA, es una empresa dedicada a la comercialización de toda clase de productos hardware. Debido a la gran competencia en el mercado y para ir acorde con las nuevas formas de comercializar productos a través de la red , se desarrolló el proyecto denominado "Sitio Web para el manejo de cotizaciones, pedidos y ofertas para la empresa PC WARE LTDA". Este pretende mejorar el servicio comercialización de productos a través de la red, más conocido como comercio electrónico o "e-comerce", implementando un sitio web o tienda virtual que ofrezca a los clientes catálogos de productos y precios. La idea es ahorrar tiempo y brindar comodidad a la gente, permitiéndoles realizar búsquedas, cotizaciones y pedidos de una manera fácil y oportuna.

El sitio agiliza y mejora el proceso de comercialización, permitiendo tener acceso a información actualizada y a tiempo, dispuesta a ser consultada por los usuarios que la soliciten.

El sistema utiliza bases de datos, programación Java y páginas JSP para la presentación de la información dinámica, permitiendo la actualización de la información a través del mismo sistema. El desarrollo del proyecto está basado en una arquitectura de 3 capas, usando componentes J2SE Java2 Standard Edition, conformada por la capa de presentación (web), capa de negocio (media) y capa de persistencia de datos (dao), usando además en el desarrollo web, el API JSP 1.2, API SERVLET 2.3, Framework Struts 1.1, y Beans.

---

* Trabajo de Grado

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Héctor Niño Quiñónez.

ABSTRACT

TITLE: WEB SITE FOR THE MANAGE OF QUOTATIONS, ORDERS AND OFFERS FOR PC WARE ENTERPRISE LTD [*]

AUTHORS: MALDONADO MANRIQUE, Sandra, y, LOPEZ POLO, Jeovany[**]

PASSWORDS: Web site, Information System, client_server application, JSP (java script page), Struts, relations data, Java, virtual shopti, e-comerce.

DESCRIPTION:

PC WARE Enterprise Ltd, is a company dedicate the marketing of hardware products. Because off great competition in the labour market and for to stay with the new form the marketing products through the Web, to develop this project called " Web Site for the manage of quotations, orders and offers for PC WARE Enterprise Ltd ", intends to improve the marketing products service through the Web "electronic commerce or e-commerce", making a Web site or a virtual shop that offers to the clients, products and prizes catalogs. The idea is to save time and to give comfort to the people, allowing to searching, quotations, and orders in a easy and at-the-time way.

The system utilize data bases, Java programming and java script page (JSP) for the information dynamic presentation, to allow update of the information, through the itself.

Devolop of the project is based in an architecture of three frames, using J2SE Java2 Standard Edition, conformed by performance frame (web) business frame (middle) and data frame (dao), using moreover in the web developing, the API JSP 1.2, API SERVLET 2.3, Framework Struts 1.1, and Beans.

With the carrying out this project intent to cover the necessity of the company and increase sale.

---

[*] Laborius the degree

[**] Physical-mechanical Ingineering Faculty. System Ingineering Acadenmic. Héctor Niño Quiñónez.

CONTENIDO

LISTA DE FIGURAS

Pág.

INTRODUCCION

El avance tecnológico que ha sufrido el mundo en los últimos años en lo que se refiere al campo de las telecomunicaciones, hace que el acceso a la información se masifique, permitiendo que las personas acudan a fuentes de información de un modo confiable y en tiempos muy cortos.

El medio más conocido, cómodo y de mayor difusión, que permite el acceso a información de diferente índole, desde cualquier parte del mundo es el *Internet,* a través de él, cualquier persona con acceso a la red, podrá consultar la información que necesiten según sus propias necesidades.

Siendo Internet, el medio de comunicación mas utilizado en los últimos años, nace la idea de realizar un proyecto aplicable a una empresa comercializadora de hardware, en este caso **PC WARE**, el cual consta del desarrollo de un sitio Web o "Tienda Virtual" que cubra necesidades de comecio, permitiendo a la empresa ofrecer sus productos hardware a través de la red, y a los clientes consultarlos en cualquier momento. Además, el sitio permite a los usuarios realizar cotizaciones y pedidos, si así lo requieren.

El presente proyecto fue desarrollado con JAVA, actualmente el lenguaje de programación más popular en Internet, disponible para el desarrollo de programas de uso general, sin olvidar que tiene un alcance completo sobre la Web.

El documento que se presenta a continuación, describe el desarrollo de los servicios de catálogos de precios, cotizaciones, pedidos y ofertas, ofrecidos por PC WARE y disponibles para que los usuarios gocen de las ventajas de la comercialización a través de la red o "Comercio Eectrónico".

1. PRESENTACIÓN

1.1 ANTECEDENTES EMPRESA PC WARE LTDA

PC WARE LTDA, es una compañía creada desde hace cinco años en Bucaramanga. Nace como un proyecto emprendedor, en el que se ha pretendido siempre, entregar al mercado un portafolio de productos muy amplio, donde el consumidor logre encontrar lo mas variado y nuevo en tecnología. PC WARE, está siempre con lo nuevo en tecnología. Ofrece productos tales como partes de computador, computadores ensamblados a la medida del usuario, periféricos, suministros, consumibles para computador, cables, servicio en redes y cableado estructurado, mantenimiento para computadores y todo lo que esto exija.

Las marcas que representan la empresa, son de importante reconocimiento mundial, como lo son: SAMSUNG ELECTRONICS, LG ELECTRONICS, MAXTOR, SEAGATE, HEWLETT PAKCARD, TOSHIBA, COMPAQ, GENIUS, QPICOM, TREDNET, CANON, SONY, LEXMARK, EPSON, AOC, ENCORE y otras que se destacan por ser pioneras en el desarrollo de nuevas tecnologías.

Hasta la fecha, la empresa PC WARE, lleva a cabo la comercialización de sus productos de forma tradicional. Por ello, para la empresa, el proyecto mas ambicioso es lograr prontamente cautivar un mercado más amplio mediante la utilización de las herramientas e-comerce en toda su extención.

La nueva forma de comercializar productos replanteada por Internet se conoce como comercio electrónico. La idea es abrir un sitio Web o “Tienda virtual” que ofrezca a través de la red, los productos que se desean dar a conocer para su venta. De esta forma, cualquier persona en cualquier momento puede acceder al sitio web, conocer precios de productos (cotizar), ofertas actuales y realizar un pedido, sin la necesidad de tener que desplazarse hasta el lugar físico del negocio, ganando el tiempo y la comodidad que la forma tradicional no ofrece.

Existen dos tipos de empresarios que abren sus tiendas virtuales en Internet. En primer lugar, se encuentran los propietarios de tiendas tradicionales que buscan

abrir nuevos mercados y dar a conocer su oferta por la red. Luego, se encuentran aquéllos que intentan la aventura de abrir directamente su cibertienda. En el primer caso, el empresario tiene solucionado el tema legal y el del stock o las existencias de productos. PC WARE desea pertenecer a éste grupo y contar con las ventajas que ofrece el comercio electrónico.

Para mejorar el proceso de comercialización de productos por parte de la empresa, se requiere el desarrollo de un Sistema Web o "Tienda Virtual", que ofrezca a través de la red sus productos y permita realizar por parte de los clientes búsquedas, cotizaciones y pedidos; además, le permita manejar ofertas especiales para clientes registrados.

EL PROYECTO PRETENDE

Mejorar el servicio de comercialización de productos por parte de la empresa PC WARE, implementando un Sitio Web o "Tienda Virtual" que dé a conocer a los clientes catálogo de productos, precios y ofertas en cualquier momento de manera oportuna.

Ahorrar tiempo y brindar comodidad a los clientes, permitiéndoles realizar búsquedas, cotizaciones y pedidos de productos, a través del sitio Web.

1.2 OBJETIVOS

1.2.1 Objetivo General

Desarrollar una aplicación que cubra las necesidades inmediatas de comercio electrónico, en el manejo de cotizaciones, pedidos y ofertas realizadas por la empresa PC WARE.

1.2.2 Objetivos Específicos

- Diseñar un modelo conceptual basado en casos de uso y su respectivo modelo físico (diagrama de entidad relación) que refleje los requerimientos del negocio de la empresa PC WARE, importantes para la aplicación a desarrollar.

- Diseñar e implementar un subsistema, que proporcione a los clientes los servicios de consultas de artículos, ofertas y realización de pedidos en línea a través de Internet.
- Desarrollo de un subsistema de administración, para los roles administrador y soporte, que permita vía Web las acciones de crear, modificar y eliminar entidades propias del negocio como lo son catálogos, artículos, ofertas y pedidos; y además permita el control de usuarios del sistema.
- Diseño de una arquitectura de tres capas para la aplicación, basada en componentes J2SE Java 2 Standard Edition (usando además el API JSP 1.2, API Servlet 2.3, framework Struts 1.1, y Beans, para la aplicación Web); conformada por la capa de presentación (capa web), capa del negocio (capa media) y la capa de persistencia de datos (capa Dao).
- Dotar la aplicación de un sistema de seguridad cuyo acceso vía Internet esté basado en roles de usuario.

1.3 IMPACTO

1.3.1 Impacto Social

Este proyecto permitirá conocer en línea las diferentes ofertas y accesorios ofrecidos por la empresa. El software podrá ser utilizado por usuarios con conexión a Internet.

1.3.2 Impacto Técnico

La aplicación estará basada en componentes implementados, utilizando la plataforma J2SE (Java 2 Standard Edition), usando además el API JSP y API Servlet (Integrados en el Tomcat) y Beans, para la aplicación Web. También se utilizará el framework Struts 1.1, el cual esta basado en el patrón MVC(Modelo-Vista-Controlador) que facilita el desarrollo de aplicaciones web fácilmente extensibles y escalables.

1.3.3 Impacto Económico

El sistema servirá como medio de apoyo en la comercialización de los diferentes artículos por parte de la empresa PC WARE, permitiendo tener un mercado más amplio y de fácil acceso gracias a Internet.

1.4 VIABILIDAD

1.4.1 Viabilidad Social

En la actualidad Internet es uno de los medios más usados para la consulta de información en línea, surgiendo así productos orientados a ésta. Este proyecto será una alternativa para que los usuarios consulten y hagan pedidos vía web desde cualquier lugar a la empresa propietaria de la aplicación.

1.4.2 Viabilidad Técnica

Existen actualmente en mercado los instrumentos y recursos técnicos necesarios para lograr la ejecución de la Proyecto, como son las herramientas tecnológicas y lenguajes de programación (Power Designer, Struts 1.1, PostgreSQL, Dreamweaver, Html, J2SE) de código abierto, que son de fácil acceso y conocimiento por parte de diferentes profesionales. Así mismo se cuenta con el conocimiento adquirido en el transcurso de nuestra carrera en el análisis y desarrollo de sistemas de información.

1.4.3 Viabilidad Económica

El uso de software gratuito o de libre distribución en el desarrollo del proyectos, hace que los costos del proyecto se reduzcan considerablemente, además de ser herramientas que se caracterizan, por su portabilidad, rápida evolución y mejoramiento continuo, como es el caso de JAVA, el cual se posiciona cada día más en el mercado y se convierte en una verdadera herramienta en el desarrollo de diferentes servicios.

1.5 DESCRIPCION GENERAL DEL DOCUMENTO

Este documento, es el resultado del desarrollo de los objetivos propuestos en el proyecto. Los sistemas informáticos diseñados e implementados, se documentaron con UML, el cual describe de manera gráfica los diferentes componentes que forman un sistema durantes las principales etapas del ciclo de vida del desarrollo de un software.

A continuación se describen los diferentes capítulos que lo componen:

El capítulo 2 contempla y describe las diversas tecnologías en las que se basa el desarrollo del proyecto. Contiene fundamentación teórica general en la que se presentan conceptos generales que se deben conocer durante el desarrollo del proyecto, y fundamentación teórica específica, que describe la filosofía y utilidad de las herramientas y tecnologías empleadas.

El capítulo 3 describe el proceso completo de desarrollo del sistema, que se llevó a cabo para lograr una implementación satisfactoria del Sitio Web y los servicios que se desean prestar.

El capítulo 4 describe el sitio Web PC WARE, mostrando su estructura como aplicación Web.

2. FUNDAMENTACION TEORICA

2.1 FUNDAMENTACION TEORICA GENERAL

2.1.1 Generalidades Internet

Internet es un conjunto de redes de ordenadores y equipos físicamente unidos mediante cables que conectan puntos de todo el mundo. Internet se inició como un proyecto de defensa de los Estados Unidos. A finales de los años 60, la ARPA (Agencia de Proyectos de Investigación Avanzados) del Departamento de Defensa definió el protocolo TCP/IP[1]. Aunque parezca extraño, la idea era garantizar mediante este sistema la comunicación entre lugares alejados en caso de ataque nuclear. Ahora el TCP/IP sirve para garantizar la transmisión de los paquetes de información entre lugares remotos, siguiendo cualquier ruta disponibe.

En 1975, ARPAnet comenzó a funcionar como red, sirviendo como base para unir centros de investigación militares y universidades, y se trabajó en desarrollar protocolos más avanzados para diferentes tipos de ordenadores y cuestiones específicas. En 1983 se adoptó el TCP/IP como estándar principal para todas las comunicaciones, y en 1990 desapareció ARPAnet para dar paso junto a otras redes TCP/IP a Internet. Por aquel entonces también comenzaron a operar organizaciones privadas de Red.
Poco a poco, todos los fabricantes de ordenadores personales y redes han incorporado el TCP/IP a sus sistemas operativos, de modo que en la actualidad cualquier equipo está listo para conectarse a Internet.

Una de las principales ventajas del protocolo TCP/IP, es que se puede modificar la ruta que sigue la información si encuentra un nodo que no funciona. Esto hace que los paquetes se transporten de forma eficiente por la red, incluso si existen nodos que estén cerrados o en reparación.

[1] TCP/IP, es un protocolo de transmisión que asigna a cada máquina que se conecta un número específico, llamado "número IP" (que actúa a modo de "número teléfono único") El protocolo TCP/IP permite describir la información en términos de paquetes, los cuales son enviados por distintos medios de transmisión como redes de cable telefónico.

Internet funciona con la arquitectura de cliente servidor, en la cual hay computadores llamados Servidores, los cuales responden a solicitudes en el momento en que se soliciten, y por otro lado existen los clientes los cuales hacen sus solicitudes.

Internet une muchas redes, se calcula que actualmente hay varios miles de redes conectadas a Internet, más de seis millones de servidores y entre 40 y 50 millones de personas que tienen acceso a sus contenidos.

2.1.1.1 World Wide Web

La World Wide Web (la "telaraña" o "maraña mundial") es tal vez el punto más visible de Internet y hoy en día el más usado junto con el correo electrónico, aunque también es de los más recientes. Originalmente denominado Proyecto WWW y desarrollado en el CERN suizo a principio de los 90, partió de la idea de definir un "sistema de hipermedios distribuidos."

La WWW puede definirse básicamente como tres cosas: hipertexto, que es un sistema de enlaces que permite saltar de unos lugares a otros; multimedia, que hace referencia al tipo de contenidos que puede manejar (texto, gráficos, vídeo, sonido y otros) e Internet, las base sobre las que se transmite la información.

El aspecto exterior de la WWW son las conocidas "páginas Web." Las páginas de la WWW están situadas en servidores de todo el mundo (sitios Web), y se accede a ellas mediante un programa denominado "navegador" (browser). Este programa emplea un protocolo llamado HTTP, que funciona sobre TCP/IP, y que se encarga de gestionar el aspecto de las páginas y los enlaces.

Los recursos apuntados desde el enlace de la página Web pueden ser de cualquier tipo:

- Otro documento Hipertextual
- Ordenadores anfitriones para hacer telnet (directorios, bases de datos)

- Ficheros para traer por FTP[2]
- Ficheros con imágenes, sonidos, videos digitalizados.
- Documentos con formato (World, Adobe, Acrobat...)

2.1.1.2 Exploradores y Servidores Web

Puesto que es un servicio de Internet, el World Wide Web está basado en clientes y servidores. Un cliente de World Wide Web se llama explorador Web o simplemente explorador, y un servidor de World Wide Web se llama servidor de red o solamente servidor. Los servidores y exploradores de World Wide Web utilizan un conjunto de reglas de comunicación llamadas Protocolo de Transferencia de Hipertexto (HTTP)[3]

Un explorador es un programa que recorre el World Wide Web y presenta páginas. El esplorador solicita una página de un servidor con base a su dirección de Internet, recupera el documento del servidor y presenta el contenido.

2.1.2 Sistemas de Información

Es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio, es el conjunto total de procedimientos, operaciones, funciones y difusión de datos o información en una organización.

Un Sistema de Información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información. A continuación se definirán cada una de estas actividades.

Entrada de Información: Es el proceso mediante el cual, el Sistema de Información toma los datos que requiere para procesar la Información. Las entradas pueden ser manuales o automáticas. Así, un Sistema de Control de Clientes podrá tener una interfase automática d entrada con el Sistema de facturación, ya que toma las facturas que genera o elabora el Sistema de Control de Clientes. Las unidades de entrada de datos a las computadoras son las

[2] FTP (File Transfer Protocol) es un protocolo que hace posible la transferencia de archivos entre computadoras.
[3] (HTTP) HyperText Transfer Protocol, es un protocolo que permite recuperar en forma rápida y efectiva, documentos hipermedia de la WWW.

terminales, cintas magnéticas, unidades de lectura de disquettes, códigos de barras, escáners, teclado, ratón, dispositivos de reconocimiento de vos, entre otros.

Almacenamiento de Información: El almacenamiento es una de las tareas o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sesión o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos. La unidad típica de almacenamiento son los disco duros, discos flexibles y los discos compactos (CD-ROM).

Procesamiento de Información: Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que estan almacenados. Esta característica de los sistemas permite la transformación de datos fuente, en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, por ejemplo que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados o un balance general de un año base.

Salida de Información: Es la capacidad de un Sistema de Información, para presentar la información procesada o bien, datos de entrada al exterior. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada de otro Sistema de información o módulo. En este caso, también existe una interfase automática de salida.

2.1.2.1 Ciclo de Vida de un Sistema de Información

El concepto de ciclo de vida de un sistema de información es medular en las investigaciones de sistemas. Durante su desarrollo, cada sistema se mueve a través de varias fases de un ciclo de vida, después del cual sólo funciona por varios años con un mínimo de mantenimiento. El sistema se deteriora gradualmente hasta el punto en que cesa de funcionar por completo y se comienza un nuevo ciclo de vida con el desarrollo de un nuevo sistema.

El método de ciclo de vida para el desarrollo de sistemas es el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implantar un sistema de información. Este método consta de las siguientes actividades:

- Investigación Preliminar
- Determinación de requerimientos del sistema
- Diseño del sistema
- Desarrollo de software
- Prueba de los sistemas
- Implementación, integración y evaluación de los sistemas

Investigación Preliminar: Este proceso se inicia con la petición o solicitud de una persona (administrador, empleado o especialistas en sistemas) para recibir ayuda de un sistema de información. La investigación preliminar se divide en tres faces:

- Aclaración de la solicitud
- Estudio de factibilidad
- Aprobación de la solicitud

Determinación de los requerimientos del sistema: En este paso es importante conocer todas las facetas de la empresa que se encuentra bajo estudio. Por esta razón, es que éste proceso de adquirir información se denomina con frecuencia: investigación detallada. Se deben estudiar los diferentes procesos de una empresa para dar respuesta a las siguientes preguntas:

- ¿Qué es lo que se hace?
- ¿Cómo se hace?
- ¿Con qué frecuencia se presenta?
- ¿Qué tan grande es el volumen de transacciones?
- ¿Cuál es el grado de eficiencia con que efectúan las tareas?
- ¿Existe algún problema?
- ¿Qué tan serio es el problema?
- ¿Cuál es la causa que lo origina?

Para contestar estas preguntas el analista conversa con varias personas de la empresa, reune detalles hacerca de los procesos de la empresa, escucha opiniones sobre los incovenientes de la misma y sobre las posibles soluciones e ideas para cambiar o mejorar el proceso.

Diseño del Sistema: Se inicia el proceso identificando los reportes y salidas que debe producir el sistema. Hecho lo anterior se determina con precisión los datos específicos para cada reporte o salida.

El diseño de un sistema también indica los datos de entrada, aquellos que serán calculados y los que deben ser almacenados. Así mismo se describe con detalle los procedimientos de cálculo y los datos individuales. Los diseñadores seleccionan las estructuras de archivo y los dispositivos de almacenamiento, tales como discos y cintas magnéticas o incluso, archivos de papel. Los procedimietos que se escriben indican, cómo procesar los datos y producir salidas. Los documentos que contienen las especificaciones de diseño representan a éste de muchas maneras: diagramas, tablas, y símbolos especiales.

La información detallada del software es proporcionada al equipo de programación para comenzar la face de desarroolo del software. Los diseñadores son los responsables de dar a los programadores, las especificaciones de software completas y claras.

Desarrollo de Software: Los encargados de desarrollar software pueden instalar (o modificar y después instalar) software comprado a terceros o escribir programas diseñados a la medida del solicitante. La elección depende del costo de cada altenativa, el tiempo disponible para escribir el software y de la disponibilidad de los programadores.

Los programadores, también son los responsables de la documentación de los programas y de proporcinar una explicación de cómo y por qué, ciertos procedimientos se codifican en determinada forma. La documentación es esencial para probar el programa y llevar a cabo el mantenimiento una vez que la aplicación se encuentre instalada.

Implantación del Sistema: La implantación es el proceso de verificar e instalar nuevos equipos, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos, necesarios para utilizarla.

Dependiendo del tamaño de la organización que empleará la aplicación y el riesgo asociado con su uso, puede elegirse comenzar la operación del sistema sólo en un área de la empresa (prueba piloto), por ejemplo: en un departamento o con una o dos personas. Algunas veces se deja que los dos sistemas, el viejo y el nuevo, trabajen en forma paralela con la finalidad de comparar los resultados. Cada estrategia de implantación tiene sus méritos de acuerdo con la situación que se considere dentro de la empresa. Sin importar cuál es la estrategia utilizada, los encargados de desarrollar el sistema procuran que el uso inicial del sistema se encuentre libre de problemas.

Integración de Sistemas: La información de los sistemas de información tendrán que diseñarse con un acoplamiento más estrecho entre la oficina y la planta. A decir verdad, el sistema de información llegará a ser tan importante en las plantas de la fabrica como en la oficina.

La tecnología informática estará enlazada en la organización para una sincronización completa y una coordinación de las operaciones. El sistema ya no estará separado funcionalmente, ni tampoco del lugar de trabajo. Este diseño dará por resultado una malla de información para la organización.

Evaluación de Sistemas: La evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes. La evaluación ocurre a lo alrgo de cualquiera de las siguientes dimensiones:

- *Evaluación operacional:* Valoración de la forma en que funciona el sistema, incluyendo su facilidad de uso, tiempo de respuesta, lo adecuado de los formatos de información, confiabilidad global y nivel de utilización.
- *Impacto organizacional:* Identificación y medición de lsobeneficios para la organización en áreas tales como finanzas (costos, ingresos y ganancias), eficiencia operacional e impacto competitivo. También se incluyen el impacto sobre el flujo de información interno y externo.

- *Opinión de los administradores:* Evaluación de las actitudes de los directivos y administradores dentro de la organización, así como de los usuarios finales.
- *Desempeño del desarrollo:* La evaluación del proceso de desarrollo de acuerdo con criterios tales como tiempo y esfuerzo de desarrollo, concuerdan con presupuestos y estándares, y otros criterios de administración de proyectos. También se incluye la valoración de los métodos y heramientas utilizados en desarrollo.

2.1.3 Bases de Datos

Una Base de Datos es el conjunto de datos almacenados con una estructura lógica. Es decir, así como son importantes los datos, la estructura conceptual con la que ellos se relacionan también los es. En la práctica, podemos pensar esto como el conjunto de datos más los programas (o *software*) que hacen de ellos un conjunto consistente.
Si no tenemos los dos factores unidos, no podemos hablar de una base de datos, ya que ambos combinados dan la coherencia necesaria para poder trabajar con los datos de una manera sistemática.

Una base de datos contiene datos para varios usuarios, donde cada uno puede tener acceso a todo el conjunto o solo a alguna parte de ellos, dependiendo de la necesidad, los cual indica que los datos son compartidos y se tiene un control centralizado sobre ellos.

Las siguientes son las definiciones necesarias para manejar con claridad y comprender el resto de los conceptos en las bases de datos relacionales:

Bases de datos relacionales.

- Operan sobre archivos o tablas de datos y no sobre datos individuales contenidos en los archivos.
- Las tablas son medios de representar la información de forma compacta y de fácil acceso.
- Estas tablas están formadas por filas y columnas.

- Las filas de una tabla equivalen a los registros, contienen los valores de los objetos o entidades descritas, y las columnas son a los campos, atributos de los objetos o entidades descritas.

Tupla : es una hilera o fila en una tabla.

Atributo: es una columna en una tabla.

Dominio: es el conjunto de valores de los cuales los atributos obtienen sus valores. Podemos también decir que un dominio es un conjunto de valores escalares del mismo tipo, dónde un valor escalar es la mínima unidad semántica de información en el sentido de que son valores atómicos.

Llave: es un atributo con una característica de relevancia para identificar la tupla.

Llave primaria: Es una llave con valores únicos, es decir, no ocurren más de una vez en el atributo.

Cardinalidad: es el número de tuplas en una tabla.

Grado: es el número de atributos en una tabla.

Relación: La definición canónica es que una relación es el producto cartesiano de dos o varios dominios.

Entidad: Es una unidad de datos en una relación con un conjunto finito de atributos. Es también conocido como n-ada, a raíz de que consiste de n-valores, uno por cada atributo.

Tabla base: es una relación autónoma a diferencia de las vistas y las tablas intermedias construidas a partir de una consulta.
Vista: es una relación virtual, que se construye a partir de tablas base o incluso otras vistas, formada por atributos de estas otras tablas de forma directa o como resultado de una consulta.

2.1.3.1 Componentes de una Base de Datos

Un sistema de Bases de Datos se compone de cinco elementos principales: datos, hardware, software, administrador y Usuarios.

Datos: Cada uno de los procesos que constituyen una organización, genera datos que son registrados en algún medio de almacenamiento que puede ser impreso, fílmico,electrónico o magnético.
Partiendo de lo particular a lo general, estos datos se pueden clasificar de la siguiente forma gerárquica:

Campo: Es la unidad más pequeña de información que se almacena en una base de datos. Permite definir una característica (edad, peso, estatura, etc.) acerca de un elemento que es objeto de estudio.

Registro: Es una colección de campos asociados (se refieren a un ente común) que permiten agrupar características acerca de un elemnto objeto de estudio.

Tabla: Es una colección de registros que contienen la información de un elemnto objeto de estudio.

Base de datos: El conjunto de estas tablas o entidades relacionadas de una forma lógica es los que se conoce como Base de Datos.

Hardware: Este se refiere a los medios de almacenamiento (discos duros, disquettes, CDs, etc.) en los cuales reside la Base de Datos, y a los dispositivos con los que manejan tales medios.

Software: Entre la base física y los denominados usuarios, existe un conjunto de programas que maneja todo acceso a la base de datos o interfaz conocida con el nombre de Sistema administrador de Bases de Datos D.B.M.S (Database Magnament System), quien es el encargado de atender los accesos de los usuarios a la base de datos. Conceptualmente lo que sucede es lo siguiente:

- Un usuario Solicita acceso, empleando algún sublenguaje de datos determinado (por ejemplo SQL).

- EL DBMS interpreta esa solicitud y la analiza.
- El DBMS inspecciona, en orden, el esquema externo[4] de ese usuario, la correspondencia externa/conceptual[5] asociada, el esquema conceptual[6], la correspondencia conceptual interna[7], y la definición de la estructura de almacenamiento.
- El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada.
- El DBMS soporta múltiples usuarios, permite la definición de datos, la manipulación de datos, la recuperación y concurrencia de datos, la seguridad de los datos y traduce en ordenes sobre la base de datos, todos los requerimientos que el usuario posee para el manejo de la información.

Administrador de bases de datos (dba): El administrador de datos es la persona que toma las decisiones estratégicas y de política con respecto a la información de la empresa, y el administrador de bases de datos (DBA) es quien proporciona el apoyo técnico necesario para poner en práctica esas decisiones. Por lo tranto el DBA está encargado del control general del sistema en el nivel técnico.

Usuario: Son todas aquellas personas que accesan la base de datos para obtener información util, la cual ayudará a resolver problemas, dar respuesta a interrogantes o cualquier otra necesidad.

2.1.3.2 Manejador de Bases de Datos (dbm)

El DBM (Data Base Manager), es la porción más importante del software de un sistema de base de datos. Es un módulo de programa que proporciona la interfase entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y las consultas hechas al sistema.

---

[4] Esquema externo: definiciones de cada uno de los diversos tipos de registros externos en cada vista externa.
[5] Correspondencia externa/conceptual: Correspondencia que existe entre una determinada vista externa y la vista conceptual.
[6] Esquema conceptual: Es la definición de una vista conceptual(es una de las vistas del contenido total de la base de datos).
[7] Correspondencia conceptual/interna: Es la que existe entre la vista conceptual y la base de datos almacenada.

Facilita las funciones de:

- almacenar físicamente
- garantizar consistencia
- garantizar integridad
- atomicidad transaccional
- manejar vistas a la información

El DBM es responsable de las siguientes tareas:

- interactuar con el manejador de archivos
- implantación de la integridad
- implantación de seguridad
- Copia de seguridad y recuperación

2.1.3.3 Lenguaje de Manipulación de Datos (dml)

Lenguaje de manipulación de datos - (DML) es un lenguaje que capacita a los usuarios a acceder o manipular datos según estén organizados por el modelo de datos.

Por manipulación se entiende:

- recuperar datos (query)
- insertar datos (insert)
- suprimir datos (delete)
- modificar datos (update)

Hay dos tipos de DML:

- *Procedural:* el usuario especifica los datos que necesita y cómo obtenerlos.
- *Nonprocedural:* el usuario especifica los datos que necesita sin especificar cómo obtenerlos.

Los datos sin procesar se almacenan usando el sistema de archivos que normalmente proporciona el sistema operativo. El DBM traduce las instrucciones de DML a mandatos del sistema de archivos de bajo nivel. El DBM es responsable del verdadero almacenamiento, recuperación y actualización de los datos de la base de datos.

2.1.3.4 Modelo de Datos

Es una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia.

De acuerdo a Ullman :
``Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. Es formal pues los objetos del sistema se manipulan siguiendo reglas perfectamente definidas y utilizando exclusivamente los operadores definidos en el sistema, independientemente de lo que estos objetos y operadores puedan significar.''
Según Codd:
``Un modelo de datos es una combinación de tres componentes:
1) una colección de estructuras de datos (los bloques constructores de cualquier base de datos que conforman el modelo);
2) una colección de operadores o reglas de inferencia, los cuales pueden ser aplicados a cualquier instancia de los tipos de datos listados en (1), para consultar o derivar datos de cualquier parte de estas estructuras en cualquier combinación deseada;
3) una colección de reglas generales de integridad, las cuales explícita o implícitamente definen un conjunto de estados consistentes --estas reglas algunas veces son expresadas como reglas de insertar-actualizar-borrar.''

Un modelo de datos puede ser usado de las siguientes maneras:
i) como una herramienta para especificar los tipos de datos y la organización de los mismos que son permisibles en una base de datos específica;
ii) como una base para el desarrollo de una metodología general de diseño para las bases de datos;
iii) como una base para el desarrollo de familias de lenguajes de alto nivel para manipulación de consultas (querys) y datos;
iv) como el elemento clave en el diseño de la arquitectura de un manejador de bases de datos.

El primer modelo de datos desarrollado con toda la formalidad que esto implica fue el modelo relacional, en 1969, mucho antes incluso que los modelos

jerárquicos y de red. A pesar de que los sistemas jerárquicos y de red como software para manejar bases de datos son previos al modelo relacional, no fue sino hasta 1973 que los modelos de tales sistemas fueron definidos, apenas unos cuantos años antes de que estos sistemas empezaran a caer en desuso.

2.1.3.5 Implementación de Bases de Datos

La conversión del modelo conceptual al modelo físico depende de la clase de administrador de la base de datos que se haya seleccionado. Las bases de datos se pueden clasificar de acuerdo con la forma en que se modelaron los datos. Estos modelos pueden ser: Entidad-Relación ampliado, Objeto semántico, Jerárquico y en Red. Cada uno de estos modelos puede ser considerado como lentes a través de los cuales observan los que desarrollan bases de datos[8].

2.1.3.6 Modelo Entidad Relación

El Modelo Entidad-Relación, es una técnica que se utiliza en la mayoría de las metodologías de desarrollo de Sistemas de Información, para que la estructura lógica global de una base de datos puede expresarse gráficamente por medio de un diagrama E-R.
Uno de los grandes atractivos del enfoque relacional es la simplicidad de los archivos y la potencia, capacidad y facilidad de integración de otros enfoques de bases de datos.

2.2 FUNDAMENTACION TEORICA ESPECIFICA

2.2.1 Uml

El lenguaje Unificado de Modelado (UML, Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de sofware.

[8] KROENKE, David M. Procesamiento de bases de datos. Fundamentos, Diseño e Instrumentación.

UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como las cosas conceptuales, tales como las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes sofware reutilizables.

UML[9] puede ser utilizado por cualquier persona involucrada en la producción, el despliegue y el mantenimiento de sofware. UML está pensado principalmente para sistemas con gran cantidad de sofware. Ha sido utilizado de forma efectiva en dominios tales como:

- Sistemas de información de empresa
- Bancos y servicios financieros
- Telecomunicaciones
- Transporte
- Defensa/industria aeroespacial
- Comercio
- Electrónica médica
- Ambito científico
- Servicios distribuidos basados en la Web

El Modelado

Es una técnica de ingeniería probada y bien aceptada. Si se quiere construir un software considerablemente grande, el problema es algo más que escribir grandes cantidades de sofware. De hecho, el truco está en crear el sotware apropiado y en escribir menos software. Esto convierte el desarrollo del software de calidad en una cuestión de arquitectura, proceso y herramientas.
Existen muchos elementos que contribuyen a que una empresa de sofware realicen y culminen sus proyectos sofware con éxito; uno en común es el uso del modelado.

[9] Información actual de UML puede encontrarse en la dirección www.rational.com y www.omg.org

Un Modelo es una simplificación de la realidad. Un modelo proporciona los planos de un sistema. Los modelos pueden involucrar planos detallados, así como planos más generales que ofrecen una visión global del sistema en consideración. Un buen modelo incluye aquellos elementos que tienen una gran influencia y omite aquellos elementos menores que no son relevantes para el nivle de abstracción dado.

Todo sistema puede ser descrito desde diferentes perspectivas utilizando diferentes modelos, y cada modelo es por lo tanto una abstacción semánticamente cerrada del sistema.

Un modelo puede ser estructural destacando la organización del sistema, o puede ser de comportamiento, resaltando la dinámica.

2.2.1.1 Modelo Conceptual de UML

Para comprender UML se necesita adquirir un modelo conceptual del lenguaje, y esto requiere aprender tres elementos principales:

- Los bloques básicos de construcción de UML
- Las reglas que dictan cómo se pueden combinar estos bloques básicos
- Algunos mecanismos comunes que se aplican a través de UML.

Bloques de Construcción UML

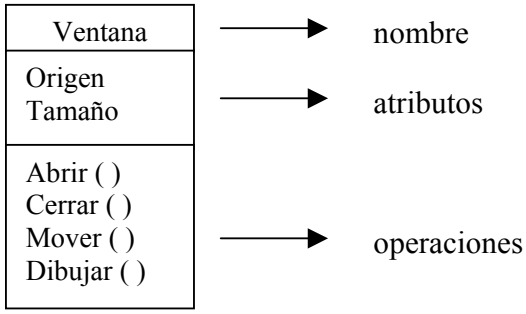
Incluye tres clases de bloques de construcción:

- Elementos
- Relaciones
- Diagramas

Elementos en UML: Hay cuatro tipos de elementos en UML. Estos elementos son los bloques básicos de construcción orientados a objetos de UML. Se utilizan para escribir modelos bien formados.

- Elementos estructurales: Son los nombres de los modelos de UML. En su mayoría son partes estáticas de un modelo, y representan cosas que son conceptuales o materiales. Hay 7 tipos de elementos estructurales:

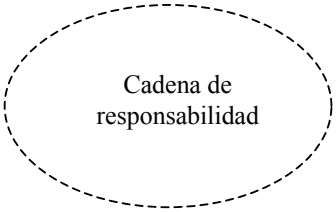
- *Clase:* Es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Una clase implementa una o más interfaces. Se representa de la siguiente manera:



- *Interfaz:* Es una colección de operaciones que especifican un servicio de una clase o componente, define un conjunto de especificaciones de operaciones, pero nunca un conjunto de implementaciones de operaciones. Se representa de la siguiente manera:

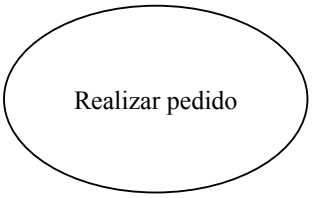
IOrtografia

- *Colaboración:* Define una interacción y es una sociedad de roles y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos. Gráficamente se representa así:

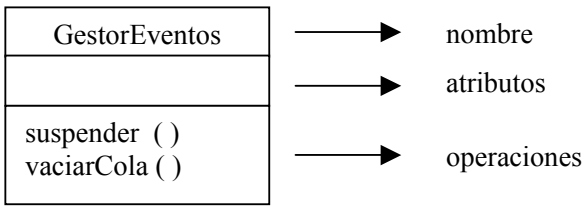


- *Caso de Uso:* Es una descripción de un conjunto de secuencias de acciones que un sistema ejecuta y que produce un resultado observable de interés para un actor particular. Se utiliza para estructurar los aspectos de

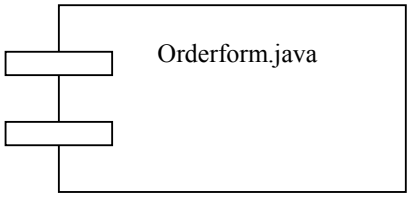
comportamiento en un modelo. Gráficamente se representa de la siguiente manera:



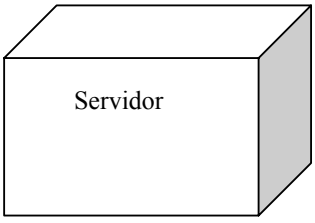
- *Clase activa:* Clase cuyos objetos tienen uno o más procesos o hilos de ejecucuón y por lo tanto pueden dar origen a actividades de control. A diferencia de una clase sus objetos representan elementos cuyo comportamiento es concurrente con otros elementos. Se representa así:



- *Componente:* Es una parte física y reemplazable de un sistema que conforma con un conjunto de interfaces y proporciona la implementación de dicho conjunto. Un componente representa típicamente el empaquetamiento físico de diferentes elementos lógicos, como clases, interfaces y colaboraciones. Gráficamente se representa así:

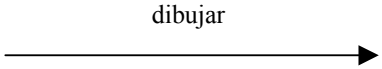


- *Nodo:* Elemento físico que existe en tiempo de ejecución y representa un recurso computacional. Un conjunto de componentes puede residir en un nodo y puede también migrar de un nodo a otro. Se representa de la siguiente manera:



- Elementos de comportamiento: Son las partes dinámicas de los modelos UML. Estos son los verbos de un modelo, y representan comportamientos en el tiempo y el espacio. Semánticamente, estos elementos están conectado a diversos elementos estructurales como clases, colaboraciones y objetos. Hay 2 tipos de elementos de comportamiento.

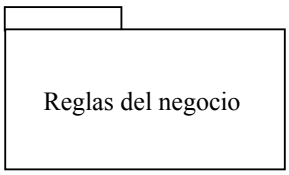
- *Interacción:* Es un comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos, dentro de un contexto particular, para alcanzar un propósito específico. Una interacción involucra muchos otros elementos, incluyendo mensajes, secuencias de acción y enlaces. Gráficamente un mensaje se muestra así:



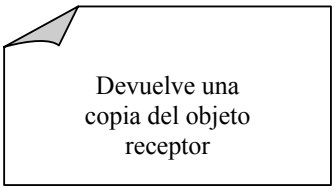
- *Máquina de Estados:* Es un comportamiento que especifica las secuencias de estados por las que pasa un objeto o una interacción durante su vida de respuesta a eventos, junto con sus reacciones a estos eventos. El comportamiento de una clase individual o una colaboración de clases puede especificarse con una máquina de estados. Una máquina de estados involucra a otros elementos incluyendo estados, transiciones, eventos y actividades. Gráficamente se representa así:

- Elementos de agrupación: Son las partes organizativas de los modelos UML. Estos son las cajas en las que puede descomponerse un modelo. Existe un solo elemento de agrupación:

- *Paquete:* Es un mecanismo de propósito general para organizar elementos en grupos. Los elementos estructurales, los elementos de comportameitno, e incluso otros elementos de agrupación pueden incluirse en un paquete. A diferencia de los componentes, un paquete es puramente conceptual. Gráficamente se representa así:

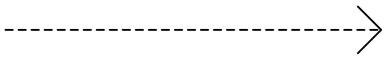


- Elementos de anotación: Son las partes explicativas de los modelos UML. Son comentarios que se pueden aplicar para describir, clasificar y hacer observaciones sobre cualquier elemento de un modelo. Hay un tipo de elemento de anotación llamado nota, que simplemente es un símbolo para mostrar restricciones y comentarios junto a un elemento o colección de elementos. Gráficamente se representa así:

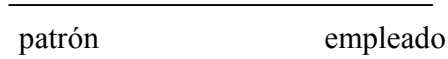


Relaciones en UML: Se utilizan para escribir modelos bien formados. Existen cuatro tipos de relaciones en UML:

- Dependencia: Relación semántica entre dos elementos, en la cual un cambio a un elemento (el elemento independiente) puede afectar a la semántica del otro elemento (el elemento dependiente). Gráficamente se representa así:

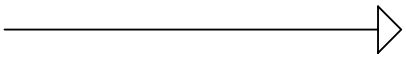


- Asociación: Es una relación estructural que describe un conjunto de enlaces, los cuales son conexiones entre objetos. La agregación es un tipo especial de asociación, que representa una relación estructural entre un todo y sus partes. Gráficamente se representa de la siguiente manera:

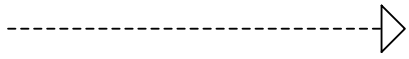


- Generalización: Es una relación de especialización/generalización, en la cual los objetos del elemento especializado (el hijo) pueden sustituir a los

objetos el elemento general (el padre). De esta forma el hijo comparte la estructura y el comportamiento del padre. Gráficamente se representa así:



- Realización: Realización semántica entre clasificadores, en donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá.Se pueden encontrar relaciones de realización en dos sitios: entre interfaces y las clases y componentes que las realizan, y entre casos de uso y las colaboraciones que los realizan. Gráficamente se representa así:



Diagramas en UML: Un diagrama es una representación gráfica de un conjunto de elementos, visializando la mayoría de las veces como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se dibujan para visualizar un sistema desde diferentesperspectivas, de forma que un diagrama es una proyección de un sistema. Para todos los sistemas, un diagrma representa una vista resumida de los elementos que constituyen un sistema. Un diagrama puede contener cualquier combinación de elementos y relaciones. UML incluye los siguientes nueve diagramas:

- Diagrama de clases: Representa un conjunto de clases, interfaces y colaboraciones, y las relaciones entre ellas. Estos diagramas son los más comunes en el modelado de sistemas orientado a objetos y se utilizan para describir la vista de diseño estática del sistema

- Diagrama de objetos: Representa un conjunto de objetos y sus relaciones. Se utilizan para describir estructuras de datos, instantáneas de las instancias de los elementos encontrados en los diagramas de clases. Estos diagramas cubren la vista de diseño estática o la vista de procesos estática de un sistema.

- Diagrama de casos de uso: Muestra un conjunto de casos de uso y actores (tipo especial de clases) y sus relaciones. Estos diagramas cubren la vista

de casos de uso estática de un sistema y son importantes en el modelado y organización del comportamiento de un sistema.

- Diagrama de secuencia: Un diagrama de secuencia es un diagrama de interacción que resalta la ordenación temporal de los mensajes. Un diagrama de interacción muestra la interacción, que consta de un conjunto de objetos y sus relaciones incluyendo los mensajes que pueden ser enviados entre ellos.

- Diagrama de colaboración: Junto con los diagramas de secuencia, un diagrama de colaboración es un diagrama de interacción que resalta la organización estructural de lso objetos que envían y reciben mensajes. Un diagrama de colaboración puede transformarse en uno de secuencia y viceversa.

- Diagrama de estados: (statechart): Muestra una máquina de estados que consta de estados, transiciones, eventos y actividades. Estos diagramas cubren la vista dinámica de un sistema y son importantes en el modelado del comportamiento de una interfaz, una clase o una colaboración.

- Diagrama de actividades: Tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema. Estos diagramas cubren la parte dinámica de un sistema. Son importantes al modelar el funcionamiento de un sistema y resalta el flujo de control entre objetos.

- Diagrama de componentes: Muestra un conjunto de componentes y sus relaciones. Se utilizan para describir la vista de implementación estática de un sistema. Estos diagramas normalmente se corresponde con una o más clases, interfaces o colaboraciones.

- Diagrama de despliegue: Muestra un conjunto de nodos y sus relaciones. Estos diagramas se utilizan para describir la vista de despliegue estática de una arquitectura. Normalmente un nodo incluye uno o más componentes.

Reglas de UML

Como cualquier lenguaje UML tiene un número de reglas que especifican a qué debe parecerse un modelo bien formado.

UML tiene reglas semánticas para:

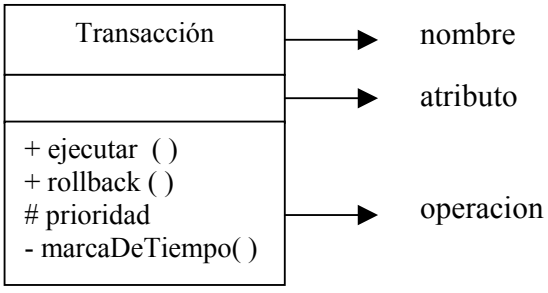
- Nombres: Cómo llamar a los elementos, relaciones y diagramas
- Alcance: El contexto que da un significado específico a un nombre
- Visibilidad: Cómo se pueden ver y utilizar esos nombres por otros
- Integridad: Cómo se relacionan apropiada y consistentemente unos elementos con otros.
- Ejecuación: Qué significa ejecutar o simular un modelo dinámico

Mecanismos Comunes UML

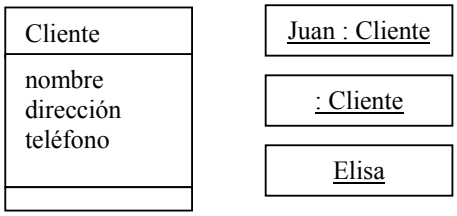
Existen cuatro mecanismos comunes que se aplican de forma consistente a través de todo el lenguaje:

- Especificaciones: En UML se utilizan para anunciar los detalles de un sistema. Proporcionan una base semántica que incluye a todos los elementos de todos los modelos de un sistema, y cada elemento está relacionado con otros de manera consistente.

- Adornos: Son los detalles que se le añaden a una clase, tales como si es abstracta o la visibilidad de sus atributos y operaciones. Estos adornos gráficos o textuales se pueden incluir en la notación rectangular básica de la clase.
- El siguiente gráfico muestra una clase con adornos que indica que es una clase abstracta con dos operaciones públicas, una protegida y otra privada.

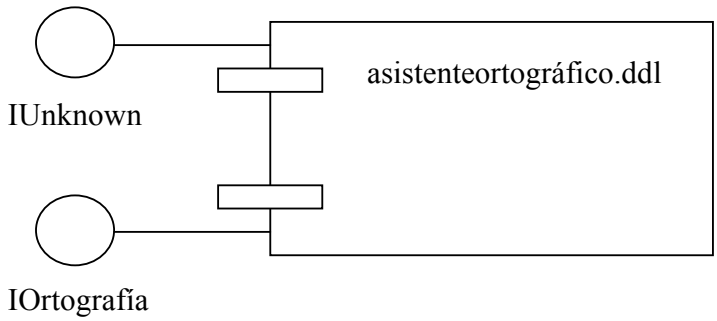


- Divisiones comunes: En primer lugar está la división entre clase y objeto. Una clase es una abstracción; un objeto es una manifestación concreta de esa abstracción. En UML se pueden modelar tanto clases como objetos, como muestra la figura:



- UML distingue un objeto utilizando el mismo símbolo de la clase subrayando el nombre del objeto.

- En segundo lugar, está la separación entre la interfaz e implementación. Una interfaz declara un contrato, y una implementación representa una realización concreta de ese contrato, responsable de hacer efectiva de forma fidedigna la semántica completa de la interfaz. En UML se pueden modelar las interfases y sus implementaciones, como se muestra en la figura:



- Mecanismos de extensibilidad: UML es un lenguaje abierto-cerrado, siendo posible configurar y extender el lenguaje de manera controlada, para las necesidades de un proyecto. Los mecanismos de extensión de UML incluyen:

- *Estereotipos:* Un estereotipo extiende el vocabulario de UML, permitiendo crear nuevos tipos de bloques de construcción que deriven de los existentes pero sean específicos a un problema.

- *Valores Etiquetados:* Un valor etiquetado extiende las propiedades de un bloque de construcción UML, permitiendo añadir nueva información en la especificación de ese elemento.

- *Restricciones:* Una restricción extiende la semántica de un bloque de construcción de UML, permitiendo añadir nuevas reglas o modificar las existentes.

2.2.1.2 Modelo Estructurado Básico

CLASES

Las clases son los bloquesde construcción más importantes de cualquier sistema orientado a obajetos. Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Las clases se pueden utilizar para capturar el vocabulario del sistema que se está desarrollando. Las clases se pueden utilizar para representar cosas que sean software, hardware o puramente conceptuales.

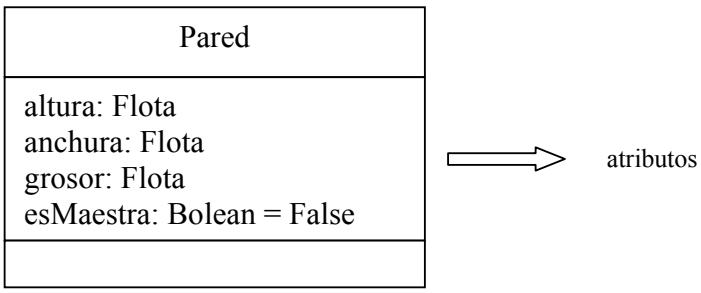
Términos y Conceptos

Una clase se representa gráficamente como un rectángulo y consta de un nombre, unos atributos y unas operaciones.

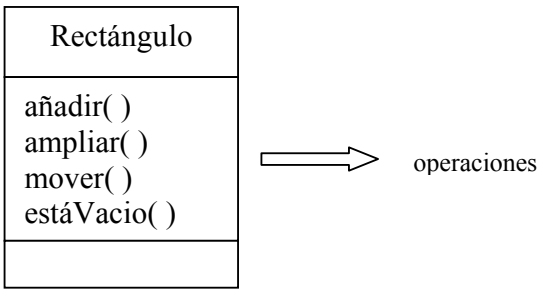
- Nombre: Cada clase debe tener un nombre que la distinga de otra clase. Un nombre es una cadena de texto. Un *nombre de camino* consta del nombre de la clase precedido por el nombre del paquete en el que se encuentra.

- Atributos: Es una propiedad de una clase identificada con un nombre, que describe un rango de valores que pueden tomar las instancias de la

propiedad. Una clase puede tener puede tener cualquier número de atributos o no tener ninguno. Un atributo representa alguna propiedad del elemento que se está modelando que es compatida por todos los objetos de esa clase. Un atributo se puede especificar más indicando su clase y quizá su valor inicial por defecto, como se muestra en la figura:



- Operaciones: : Una operación es la implementación de un servicio que puede ser requerido a cualquier objeto de la clase para que muestre un comportamiento. En otras palabras, una operación es una abstracción de algo que se puede hacer a un objeto y que es compartido por todos los objetos de una clase. Las operaciones se pueden representar sólo ostrando sus nombres como se muestra en la figura:



RELACIONES

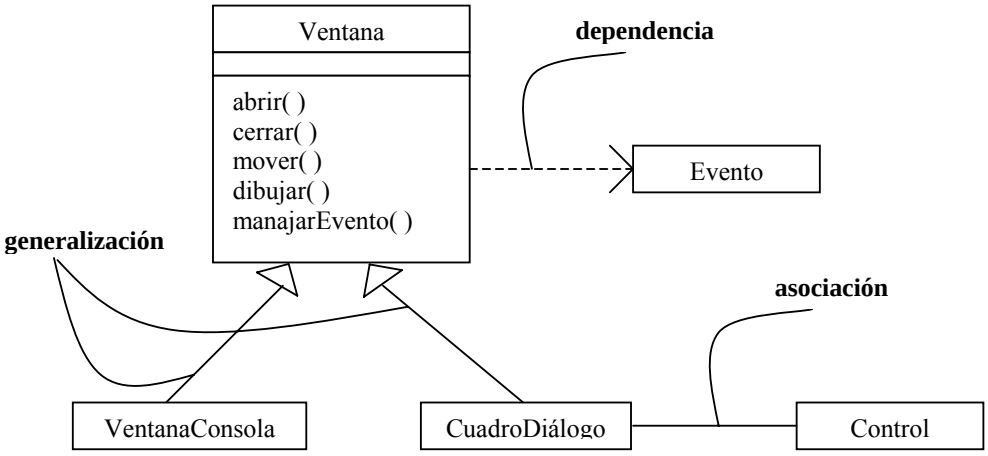
Al modelar un sistema, se debe tener en cuenta, que no sólo hay que identificar los elementos que conforman el vocabulario del sistema, también hay que modelar cómo se relacionan estos elementos entre sí.

En el modelado orientado a objetos hay tres tipos de relaciones especialmente importantes:

- Dependencias
- Generalizaciones
- Asociaciones

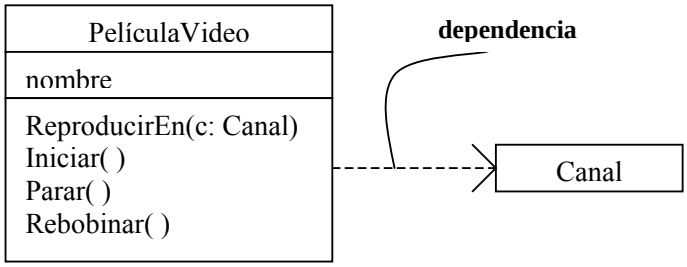
Términos y Conceptos

Una relación es una conexión entre elementos. El gráfico que se muestra a continuación muestra cada uno de los tipos de relaciones.



Relaciones

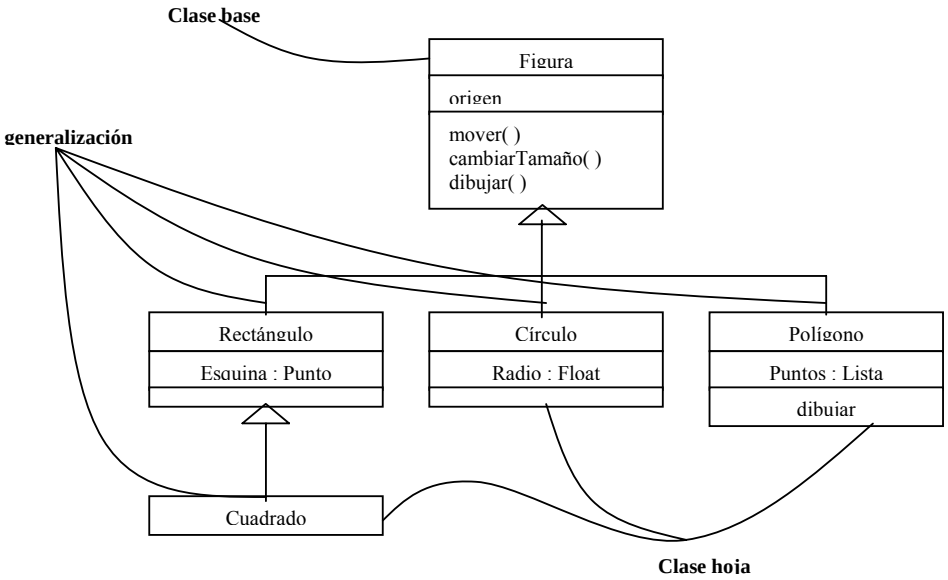
- Dependencia: Una dependencia es una relación de uso que declara que un cambio en la especificación de un elemento (por ejemplo la clase Evento), puede afectar a otro elemento que la utiliza (por ejemplo la clase Ventana), pero necesariamente a la inversa. Las dependencias se usarán cuando se quiera indicar que un elemento utiliza otro.



Dependencias

- Generalización: Una generalización es una relación entre un elemento general (llamado superclase o padre) y un caso más específico de ese elemento (llamado subclase o hijo). La generalización significa que los

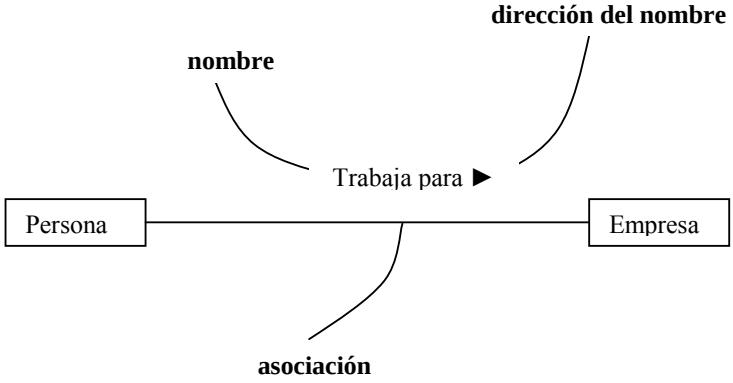
objetos hijos se pueden emplear en cualquier lugar que pueda aparecer el padre, pero no a la inversa.



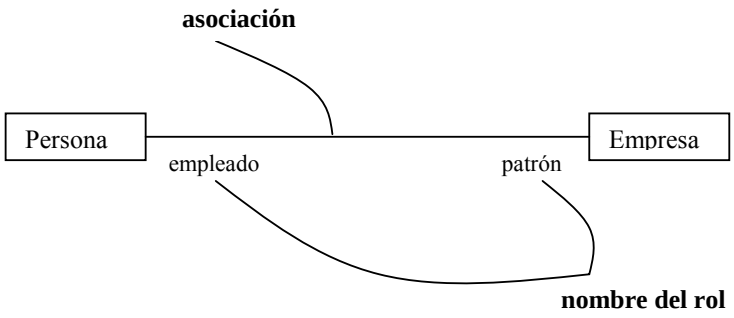
Generalización

- Asociación: Es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro. Dada una asociación entre dos clases, se puede navegar desde un objeto de una clase hasta un objeto de la otra clase, y viceversa. Es legal que ambos extremos de una asociación estén conectaods a la misma clase. Esto significa que, dado un objeto de la clase, se puede conectar con otros objetos de la misma clase.

Nombre: Una asociación puede tener un nombre que se utiliza para describir la naturaleza de la relación

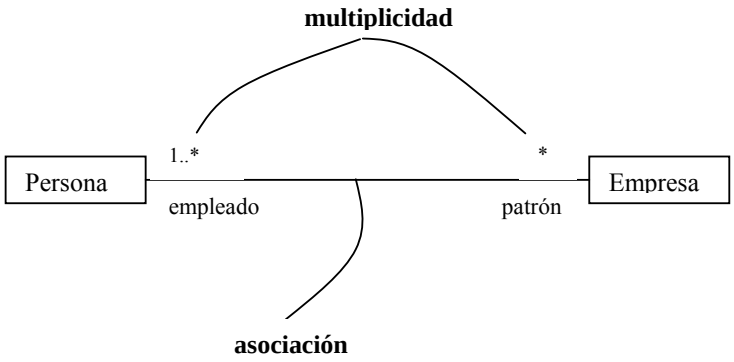


Rol: Cuando una clase participa en una asociación, tiene un rol específico que juega en la asociación; un rol es simplemente la cara que la clase de un extremo de la asociación presenta a la clase del otro extremo.



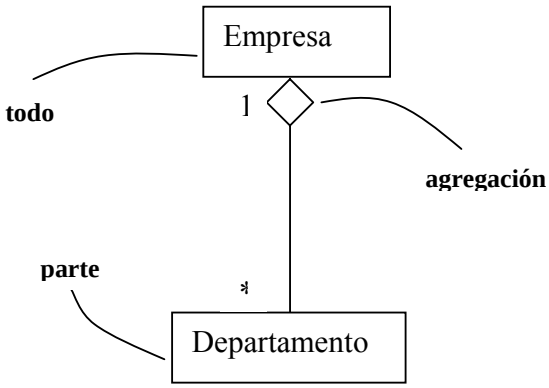
Roles

Multiplicidad: Una asociación representa una relación estructural entre objetos. En muchas situaciones de modelado, es importante señalar cuántos objetos pueden conectarse a través de una instancia de una asociación.



Multiplicidad

Agregación: A veces, se desea modelar una relación “todo/parte”, en el cual una clase representa una cosa grande (el “todo”), que consta de elementos más pequeños (“las partes”). Este tipo de relación denominado agregción representa una relación deltipo “tiene un “, osea, un objeto del todo tiene objetos de la parte.



Agregación

DIAGRAMAS

Con UML se construyen modelos a partir de bloques de construcción básicos, tales como clases, interfaces, colaboraciones, componentes, nodos, dependencias, generalizaciones y asociaciones. Los diagramas son los medios para ver estos bloques de construcción. Un diagrama es una representación gráfica de un conjunto de elementos.

Cuando se modelan sistemas reales, sea cual sea el dominio del proble, muchas veces se dibujan los mismos tipos de diagramas, porque representan vistas comunes de modelos comunes. Normalmente, las partes estáticas de un sistema se representan mediante diagramas estructurales.

Diagramas Estructurales: Los diagramas estructurales de UML, existen para visualizar, especificar, construir y documentar los aspectos estáticos de un sistema. Existen cuatro diagramas estructurales:

- Diagrama de clases: Clases interfaces y colaboraciones.
- Diagrama de objetos: Objetos.
- Diagrama de componentes: Componentes.
- Diagrama de despliegue: Nodos

Diagramas de comportamiento: Los diagramas de comportamiento se emplean para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema. Existen cinco diagramas de comportamiento

- Diagrama de casos de uso: Organiza los comportamientos del sistema
- Diagrama de secuencia: Centrado en la ordenación temporal de los
- Mesajes.
- Diagrama de colaboración: Centrado en la organización estructural de los
- objetos que envían y reciben mensajes.
- Diagrama de estados: Centrado en el estado cambiante de un
- sistema dirigido por eventos.
- Diagrama de actividades: Centrado en el flujo de control de actividades.

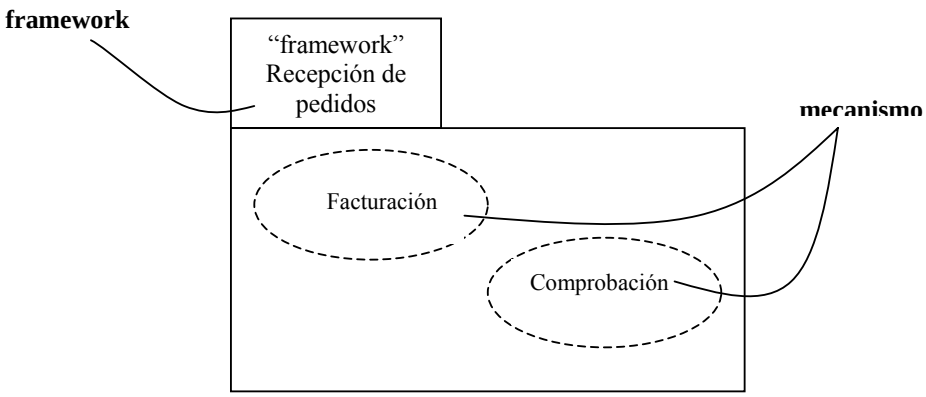
2.2.1.3 Modelado Arquitectónico

PATRONES

Todos los sistemas bien estructurados están llenos de patrones. Un patrón proporciona una solución común a un problema común en un contexto dado. Los patrones se utilizan para especificar *los mecanismos* y *frameworks* que configuran la arquitectura del sistema. Un patrón se puede hacer asequible identificando de manera clara todos los elementos variables que puede ajustar un usuario del patrón para aplicarlos a un contexto particular.

Los patrones ayudan a visualizar, especificar, construir y documentar artefactos de un sistema con gran cantidad de software.

En UML, normalmente se modelarán patrones de diseño (también llamados mecanismos), los cuales se pueden representar como colaboraciones. De forma similar, los patrones arquitectónicos se modelarán normalmente como *frameworks,* que se pueden representar como paquetes estereotipados.
UML proporciona una representación gráfica para ambos tipos de patrones, como se muestra en la siguiente figura:



Patrones de diseño (mecanismos)

Un mecanismo es un patrón de diseño que se aplica a una sociedad de clases. En otras palabras un mecanismo es tan sólo otro nombre para un patrón de diseño que se aplica a una sociedad de clases. Por ejemplo, un problema de diseño común que aparece en java es adaptar una clase que sabe cómo responder a un conjunto de eventos, de forma que resporda a un conjunto ligeramente diferente, sin alterar la clase original. Una solución es patrón adaptador, un patrón de diseño estuctural que convierte una intefaz en otra.

Framework (patrón arquitectónico)

Framework es un patrón arquitectónico que proporcionauna plantilla extensible para aplicaciones dentro de un dominio. Un framework es algo más grande que un mecanismo. De echo, se puede pensar en un framework como una microarquitectura que incluye un conjunto de mecanismos que colaboran para resolver un problema común en un dominio común. Cuando se especifica un framework, se especifica el esqueleto de una arquetectura, junto a los elementos variables, que se muestran a los usuarios que quieren adaptar el framework a su propio contexto.

En UML, un framework se modela como un paquete estereotipado. Cuando se mira dentro del paquete se pueden ver mecanismos existentes en cualquiera de las diferentes vistas de la arquitectura de un sistema.

2.2.1.4 Framework Struts

ANTECEDENTES: EL MODELO 1 Y 2

Cuando se inventaron los Servlets Java, muchos programadores se dieron cuenta de que eran una buena idea. Eran más rápidos y más potentes que el CGI estándard, eran portables, y extensibles infinitamente.

Pero escribir infinitas sentencias `println()` para enviar HTML al navegador era tedioso y problemático. La respuesta fueron las JavaServer Pages, que nos dejaron escribir servlets dentro de ellas. Ahora los desarrolladores podían mezclar fácilmente HTML con código Java, y tener todas las ventajas de los servlets.

Las aplicaciones web Java se convirtieron y centraron rápidamente en "JSP". Esto, por sí sólo no era malo, pero hacían poco por resolver problemas de control de flujo y otros problemas endémicos de las aplicaciones Web. Claramente se necesitaba otro modelo...

Muchos desarrolladores inteligentes se dieron cuenta que las JavaServer Pages y los servlets se podrían usar juntos para desplegar aplicaciones web. Los servlets podrían ayudar con el control de flujo, y las JPSs podrían enfocarse en la tarea odiosa de escribir HTML. Usar JSP y servlets juntos se ha dado ha conocer como el **Modelo 2**, cuando usar sólo JSPs era el **Modelo 1**.

El Modelo 2 de JSPs sigue el clásico patrón de diseño *Modelo-Vista-Controlador* de SmallTalk. Ahora es muy común usar los terminos Modelo 2 y MVC indistintamente.

El proyecto Struts lo lanzó Craig R. McClanahan en Mayo del 2000, para proporcionar un marco de trabajo MVC estándard a la comunidad Java. En Julio del 2001, se liberó Struts 1.0, e IOHO.

EL PATRON DE DISEÑO (MVC) MODELO VISTA-CONTROLADOR

En el patrón de diseño MVC, el flujo de la aplicación está dirigido por un Controlador central. El Controlador delega solicitudes, en nuestro caso, solicitudes HTTP, a un manejador apropiado. Los manejadores están unidos a un Modelo, y cada manejador actúa como un adaptador entre la solicitud y el Modelo. El Modelo representa, o encapsula, un estado o lógica de negocio de la aplicación. Luego el control normalmente es devuelto a través del Controlador hacia la Vista apropiada. El reenvío puede determinarse consultando los conjuntos de mapeos, normalmente cargados desde una base de datos o un fichero de configuración. Esto proporciona un acoplamiento cercano entre la Vista y el Modelo, que puede hacer las aplicaciones significativamente más fáciles de crear y de mantener.

El Modelo: La parte del Modelo de un sistema basado en MVC puede dividirse en conceptos--el estado interno del sistema, y las acciones que pueden tomarse para cambiar el estado. En términos gramáticos, podríamos pensar en la información de estado como *nombres* (cosas) y las acciones como *verbos* (cambios del estado de esas cosas). Las aplicaciones de gran escala normalmente representarán un conjunto de posibles acciones lógicas de negocio con métodos que pueden ser llamados sobre los beans que mantienen su información de estado. Por ejemplo, podríamos tener un bean de una tarjeta de compra, almacenado en el ámbito de sesión por cada usuario actual con las propiedades que representan el conjunto actual de ítems que el usuario ha decidio comprar. Este bean también podría tener un método `checkOut()` que autorice la tarjeta de crédito del usuario, y envíe el pedio al almacen para que sea remitido. Otros sistemas representarán las acciones disponibles de forma separada, quizas como Session Enterprise JavaBeans (Session EJBs). Por otro lado, en algunas aplicaciones de menor escala, las acciones disponibles podrían estar embebidas dentro de clases `Action` que son parte del rol del Controlador. Esto es apropiado cuando la lógica es muy simple, o donde no está contemplada la reutilización de la lógica de negocio en otros entornos. El marco de trabajo Struts soporta cualquiera de estas aproximaciones, pero nosotros recomendamos encarecidamente separar la lógica de negocio ("cómo se hace") del rol que juegan las clases `Action` ("que hace").

La Vista: La parte de la Vista de una aplicación basada en Struts generalmente está construida usando tecnología JavaServer Pages (JSP). Las págnas JSP pueden contener texto HTML estático (o XML) llamado "plantilla de texto", además de la habilidad de insertar contenido dinámico basado en la interpretación (en el momento de solicitud de la página) de etiquetas de acción especiales. El entorno JSP incluye un conjunto de etiquetas estándard, como `<jsp:useBean>`. Además, hay una facilidad estándard para definir nuestras propias etiquetas, que están organizadas en "librerías de etiquetas personalizadas". Struts incluye una extensa librería de etiquetas personalizadas que facilitan la creación de interfaces de usuario que están completamente internacionalizados, y que interactúan amigablemente con beans `ActionForm` que son parte del Modelo del sistema. Además de las páginas JSP y la acción y las etiquetas personalizadas que contienen, normalmente los objetos de negocio necesitan poder dibujarse a sí mismos en HTML (o XML), basándose en su estado actual en el momento de la solicitud. La salida renderizada desde dichos objetos puede incluirse fácilmente en una página JSP resultante usando la etiqueta de acción estándard `<jsp:include>`.

El Controlador: La parte Controlador de la aplicación está enfocada en las solicitudes recibidas desde el cliente (normalmente un usuario ejecutando un navegador Web), decidiendo qué función de la lógica de negocio se va a realizar, y luego delegando la responsabilidad para producir la siguiente fase del interface de usuario en un componente Vista apropiado. En Struts, el componente principal del Controlador es un servlet de la clase `ActionServlet`. Este servlet está configurado definiendo un conjunto de `ActionMappings`. Un `ActionMapping` define un `path` que se compara contra la URI solicitada de la solicitud entrante, y normalmente especifica el nombre totalmente cualificado de clase de una clase *Action*. Todas las *Actions* son subclases de `org.apache.struts.action.Action`. Las acciones encapsulan la lógica del negocio, interpretan la salida, y por último despachan el control al componente Vista apropiado para la respuesta creada. Struts también soporta la habilidad de usar clases `ActionMapping` que tienen propiedades adicionales más allá de las estándard requeridas para operar el marco de trabajo. Esto nos permite almacenar información adicional específica de nuestra aplicación, pero aún utiliza las características restantes del marco de

trabajo. Además, Struts nos permite definir nombres lógicos para los controles a los que se debería reenviar para que un método actión pueda preguntar por la página "Main Menu" (por ejemplo), sin saber el nombre real de la página JSP correspondiente. Estas características nos ayudan a separar la lógica de control (qué hacer) de la lógica de la vista (cómo se renderiza).

INTRODUCCION AL MARCO DE TRABAJO DE STRUTS

Creyendo en el patrón de diseño Modelo-Vista-Controlador, las aplicaciones Struts tiene tres componentes principales: un **servlet controlador**, que está proporcionado por el propio Struts, **páginas JSP** (la "vista"), y la lógica de negocio de la aplicación (o **el "modelo**"). Veamos como esto funciona todo junto.

El servlet controlador Struts une y enruta solicitudes HTTP a otros objetos del marco de trabajo, incluyendo JavaServer Pages y subclases `org.apache.struts.action.Action` porporcionadas por el desarrollador Struts. Una vez inizializado, el controlador analiza un fichero de configuración de recursos, La configuración de recursos define (entre otras cosas) los `org.apache.struts.action.ActionMapping` para una aplicación. El controlador usa estos mapeos para convertir las solicitudes HTTP en acciones de aplicación.

Un `ActionMapping` normalmente especificará:

- una path solicitado (o "URL"),
- El tipo objeto (subclase de `Action`) para actuar sobre la solicitud,
- y otras propiedades según se necesite.

El objeto `Action` puede manejar la solicitud y responder al cliente (normalmente un navegador Web), o indicar a que control debería ser reenviado. Por ejemplo, si un logín tiene éxito, una acción logín podría desear reenviar la petición hacia el *mainMenu.*

Los objetos `Action` tienen acceso al servlet controlador de la aplicación, y por eso tienen acceso a los métodos del servlet. Cuando se reenvia un control, un objeto *Action* puede reenviar indirectametne uno o más objetos compartidos,

incluyendo JavaBeans, situándolos en una de las colecciones estándard compartidas por los servlets Java.

Un objeto acción puede crear un bean de tarjeta de compra, o un ítem de la tarjeta, situando el bean en la colección de sesión, y luego reenviando el control a otro mapeo. Este mapeo podría usar una página JavaServer Page para mostrar los contenidos de la tarjeta del usuario. Como cada cliente tiene su propia sesión, cada uno también tendrá su propia tarjeta de compra. En una aplicación Struts, la mayoría de la lógica del negocio se puede representar usando JavaBeans. Una `Action` puede llamar a las propiedades de un JavaBean sin conocer realmente como funciona. Esto encapsula la lógica del negocio, para que la `Action` pueda enfocarse en el manejo de errores y dónde reenviar el control.

Los java bean como manejadores de formularios de entrada

Los JavaBeans también se pueden usar para manejar formularios de entrada. Un problema clave en el diseño de aplicaciones Web es retener y validar lo que el usuario ha introducido entre solicitudes. Con *Struts*, podemos definir un conjunto de clases bean formulario, subclasificando `org.apache.struts.action.ActionForm`, y almacenar fácilmente los datos de un formulario de entrada en estos beans formularios. El bean se graba en una de las colecciones estándard o de contexto compartidas, por eso puede ser usado por otros objetos, especialmente un objeto `Action`.
El bean de formulario puede usarlo una JSP para recoger datos del usuario por un objeto `Action` para validar los datos introducidos por el usuario y luego de nuevo por la JSP para rellenar los campos del fomulario. En el caso de validación de errores, *Struts* tiene un mecanismo compartido para lanzar y mostrar mensajes de error.

Un bean de formulario *Struts* se declara en la configuración de recursos definida en un fichero fuente Java, y enlazado a un `ActionMapping` usando un nombre de propiedad comnún. Cuando una solicitud llama a un *Action* que usa un bean de formulario, el servlet controlador recupera o crea el bean formulario, y lo pasa el objeto *Action*. Este objeto entonces puede chequear los contenidos del bean de

formulario antes de que su formulario de entrada se muestre, y también la cola de mensajes a manejar por el formulario. Cuando esta listo, el objeto *Action* puede devolver el control con un reenvio a su formulario de entrada, usando un JSP. El controlador puede responder a la solicitud HTTP y dirigir al cliente a la JavaServer Page.

Etiquetas personalizadas

Las etiquetas personalizadas en el marco de trabajo *Struts* están diseñadas para usar las características de internacionalización incluidas en la plataforma Java. Todas las etiquetas de campos y los mensajes pueden recuperarse desde un recurso de mensajes, y Java puede proporcionar automáticamente el recurso correcto para el idioma y país de un cliente. Para proporcionar mensajes para otro idioma, simplemente añadimos otro fichero de recurso.

Junto al internacionalismo, otros beneficios de esta aproximación son las etiquetas consistentes entre formularios, y la posibilidad de revisar todas las etiquetas y mensajes desde una localización central.

Para la aplicación más simple, un objeto *Action* podría algunas veces manejar la lógica de negocio asociada con una solicitud. Sin embargo, en la mayoría de los casos, un objeto Action, debería llamar a otro objeto, normalmente un JavaBean, para realizar la lógica de negocio real. Esto permite al objeto *Action* enfocarse en el manejo de errores y el control de flujo, en vez de en la lógica del negocio. Para permitir su reutilizacion en otras plataformas, los JavaBeans de lógica de negocio no deberían referirse a ningún objeto de aplicación Web. El objeto *Action* debería traducir los detalles necesarios de la solicitud HTTP y pasarlos a los beans de la lógica del negocio como variables normales de Java.

Por ejemplo, en una aplicación de base de datos:

- Un bean de lógica de negocio conectaría y consultaría la base de datos,
- El bean de lógica de negocio devolvería el resultado al objeto *Action*,
- El objeto *Action* almacenaría el resultado en un bean formulario en la solicitud

- La JavaServer Page mostraría el resultado en un formulario HTML.

Ni el objeto *Action* ni la página JSP necesitan saber (o no les importa) de dónde viene le resultado. Sólo necesitan saber cómo empaquetarlo y mostrarlo.

2.2.2 Programación Java

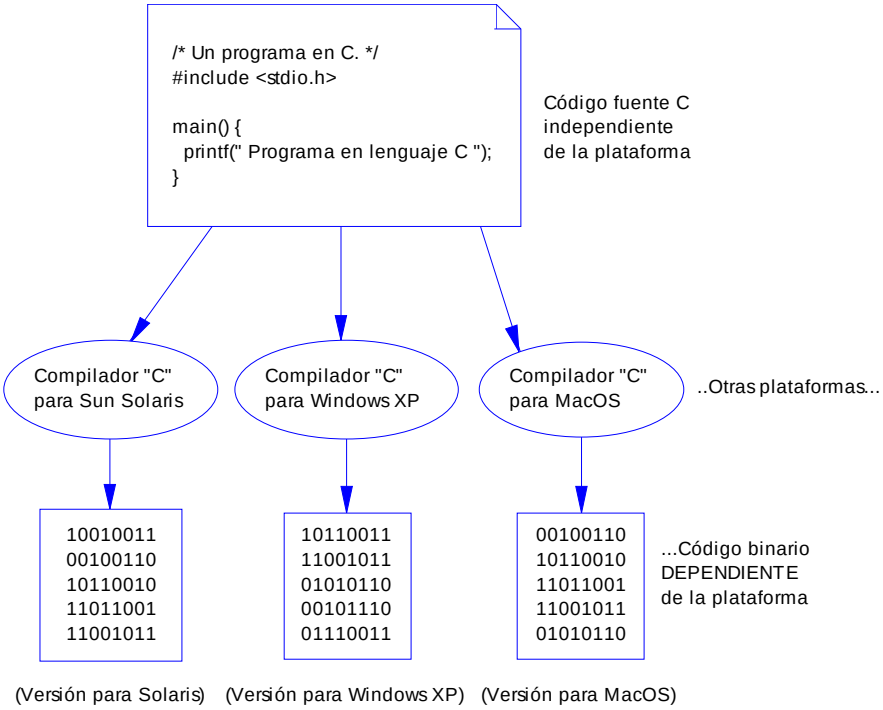
Para la programación de los servicios Web del proyecto, se usó el lenguaje de programación Java. Java surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación que fuera independiente de la plataforma, y que pudiera ser utilizado para crear software para diversos dispositivos electrónicos.

El concepto general de JAVA, es el siguiente: fue un lenguaje desarrollado desde el comienzo teniendo en mente las redes de computadores y la posibilidad de contar con un software que funcionara en muchos tipos de sistemas operativos.

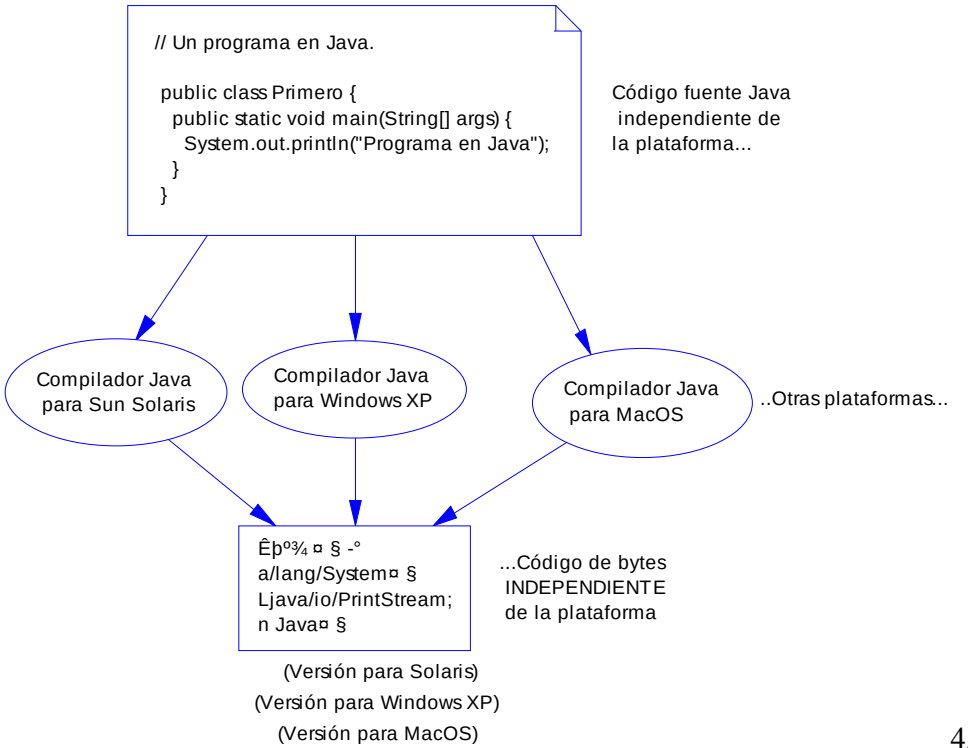
La compañía *Sun* describe el lenguaje Java como “simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portable, alto rendimiento, multihilo y dinámico”.

Como mencionábamos anteriormente, debido a la existencia de distintos tipos de CPUs y a los continuos cambios, era importante conseguir una herramienta independiente del tipo de CPU utilizada. Desarrollan un código “neutro” que no depende del tipo de electrodoméstico, el cual se ejecuta sobre una “máquina hipotética o virtual” denominada *Java Virtual Machine* (JVM). Es la JVM quien interpreta el código neutro convirtiéndolo a código particular de la CPU utilizada. Esto permitía lo que luego se ha convertido en el principal lema del lenguaje: “Write Once, Run Everywhere”.

Las siguientes figuras ilustran lo anterior:



Compiladores para diferentes plataformas



Java Virtual Machine (JVM).

2.2.2.1 Características de Java

Orientado a Objetos: Java es un lenguaje que está basado en el paradigma orientado a objetos. Esto quiere decir que el programador debe enfocarse en los datos y en métodos que manipulan esos datos, más que en procedimientos. En este lenguaje, una clase es la colección de datos y métodos que operan sobre esos datos, llamado esto encapsulamiento. Las clases se organizan de forma jerárquica de modo que existen las llamadas superclases y las subclases que "heredan" su comportamiento. También soporta el llamado "polimorfismo", el cual permite que una clase tenga métodos llamados de la misma manera y que sea el lenguaje el que seleccione cual es el que debe usar, de acuerdo con los tipos o cantidad de parámetros con el que es llamado.

Para trabajar con las clases es necesario el uso de objetos llamados instancias de la clase. Las instancias, al igual que en C++ necesitan obtener espacio en memoria mediante un constructor, y lo liberan en el momento de su destrucción. JAVA incluye gran cantidad de clases útiles para manejar los gráficos o formularios, redes funciones científicas, conectividad con bases de datos, etc.

JAVA es interpretado: A diferencia de otros programas generados en otros lenguajes, los cuales generan código nativo de la máquina en que funciona, en JAVA, estos son interpretados por la Máquina Virtual de JAVA o JVM de su sigla en inglés. Cuando se compila un programa en JAVA, se están generando "byte-codes", los cuales son independientes de la plataforma en que son generados. En el momento de la ejecución, la JVM toma estos bytes-codes y los interpreta en un lenguaje que la máquina pueda entender. Esto obliga a que la máquina que pretende ejecutar JAVA tenga instalado la JVM, sin embargo esta ha sido desarrollada para muchas plataformas como Microsoft Windows, Sun Solaris, UNIX, LINUX, MacOS, etc.

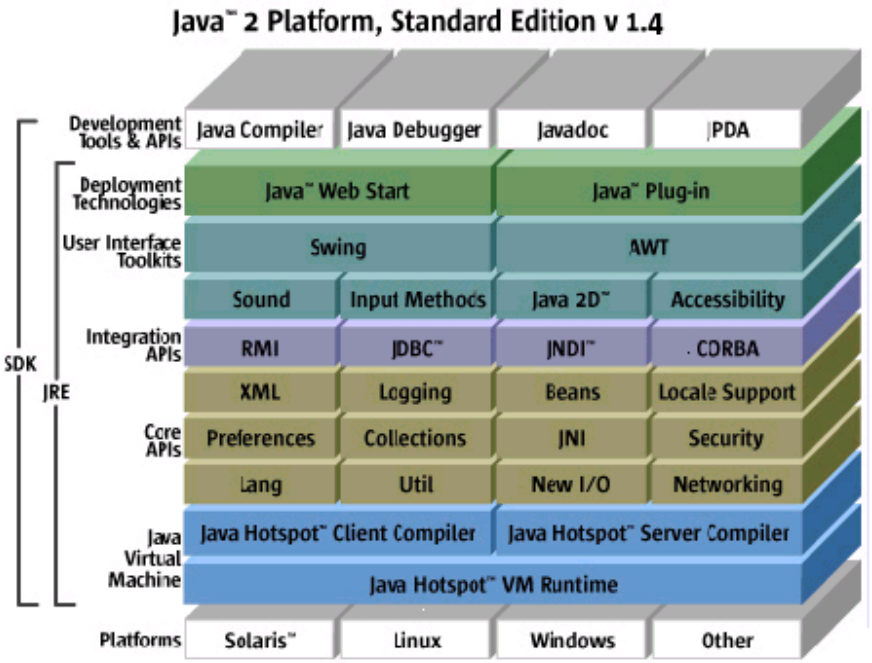
JAVA es sencillo: El lenguaje fue creado de tal modo que los programadores pudieran aprenderlo de forma fácil. Sigue las características básicas de la sintaxis del lenguaje C++, y elimina algunas características de otros lenguajes que hacen que su uso tenga mas probabilidad de error o problemas de seguridad. JAVA no tiene por ejemplo apuntadores, ni necesita la implementación de destructores, ya que la JVM cuenta con un recolector de basura, el cual elimina objetos que ya no son referenciados por los programas en ejecución, de forma dinámica.

JAVA permite la creación de programas que son altamente modulares mediante el uso de clases, lo que facilita su desarrollo y mantenimiento. Las aplicaciones que son creadas con JAVA podrán funcionar en cualquier máquina que tenga instalada la JVM, siempre que el programador no use tecnologías externas que impidan su portabilidad.

2.2.2.2 Tecnologías Java 2

Java 2 Platform, Standard Edition (J2SE): Constituye el corazón de la plataforma Java 2. La compañía *Sun*, creadora de *Java*, distribuye gratuitamente el *Java 2 SDK(Java 2 System Development Kit)*, se trata de un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en *Java*.

Figura 1. Plataforma Java2



Java 2 Platform, Enterprise Edition (J2EE): Incluye J2SE y Se ha convertido en el estándar multiplataforma para el despliegue de aplicaciones empresariales. La utilización de arquitecturas abiertas, la independencia de la plataforma y de la base de datos, la utilización de servicios web y la integración de las aplicaciones constituyen los elementos de fuerza que hacen que J2EE sea diferente.

Java 2 Platform, Micro Edition (J2ME): es una versión reducida de la plataforma Java 2 para su uso en pequeños dispositivos como teléfonos o PALM.

2.2.3 Entorno de programación Web

2.2.3.1 Lenguaje html

HTML es un lenguaje para la definición de estilos lógicos en documentos de hipertexto, siendo el medio principal para la diseminación de información en World Wide Web (www)[10]. HTML se limita a describir la estructura y el contenido de un documento, nunca el formato de la página y su apariencia, ya que éstos son muy dependientes del visualizador utilizado.

El formato de estos documentos, y la inclusión de otros elementos, se hace por medio de marcas (o etiquetas), las cuales pueden utilizar una serie de parámetros (atributos). Este código es insertado en archivos los cuales pueden ser cargados en un browser[11] en el www. Las etiquetas le dicen al Web browser, cómo desplegar palabras e imágines en una página Web para se visualizadas por el uauario.

[10] El World Wide Web (www) son todos los recursos y usuarios en Internet que usan el Protocolo de Transferencia de Hipertexto (http).
[11] Un browser es una aplicación que permite interactuar con toda la información en el World Wide Web

El HTML y WWW generalmente los incluyen los principales browsers como Microsoft Internet Explorer y Nestcape Navigator.

2.2.3.2 Protocolo http

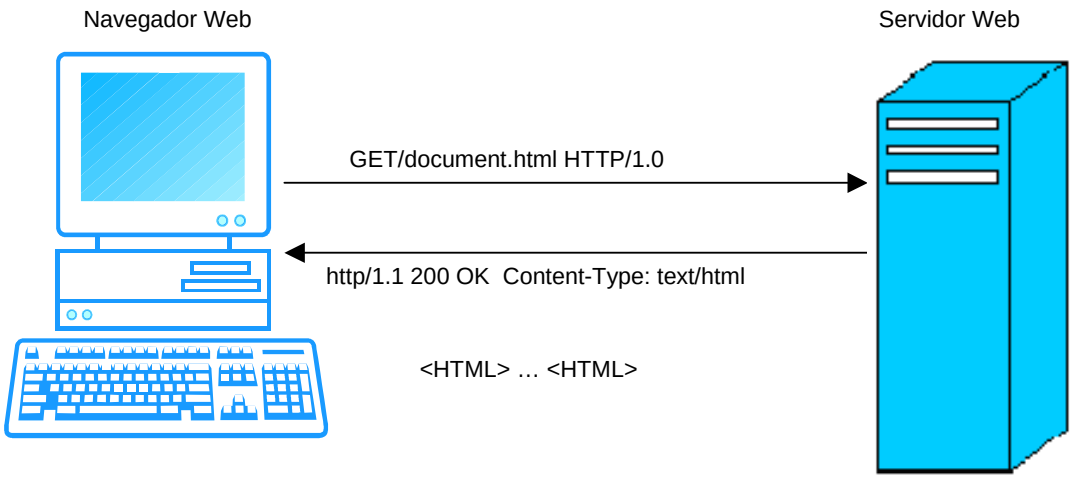
El http, es el lenguaje que se encarga de la transmisión del documento de hipertexto o página web entre el cliente y el servidor, mediante el uso de los mensajes.

http proporciona las normas para que los navegadores hagan peticiones y los servidores entreguen respuestas. Este conjunto de normas, o protocolo, incluye la manera de:

- Solicitar un documento por su nombre
- Ponerse de acuerdo en el formato de los datos
- Determinar quién es el usuario
- Decidir cómo manejar recursos obsoletos
- Indicar los resultados de una petición

Http consiste en un conjunto de comandos escritos como líneas de texto ASCII ordinario. Cuando se analiza un navegador Web, no se tiene acceso directo a los comando http. Sin embargo, al escribir una dirección URL o al hacer clic en un hiperenlace, el navegador convierte su acción en comandos http que piden el documento al servidor especificado en la URL. El servidor Web encuentra el documento y lo envía al navegador, que lo muestra al usuario.

Figura 2. Operación básica de HTTP



Petición HTTP del Servidor

Una petición o los mensajes http, tienen un formato que consta de un máximo de cuatro partes: la línea de petición o respuesta, la cabecera http, una línea en blanco que señala el final de las cabeceras y el cuerpo (body) http. El contenido de las secciones es diferente si el mensaje proviene del cliente (petición) o del servidor (respuesta).

La Primera parte es ***la línea de Petición*** o *request line:* consta de 3 categorías, separadas por espacios.

- El método de la petición. Los métodos principales en el protocolo son GET, que indica que la solicitud de información, se presenta en una URL particular. El método HEAD indica la solicitud de la cabecera http únicamente, y no más datos. El método POST indica que hay datos que serán enviados al servidos como parte del cuerpo del mensaje http.
- La URI (Identificador uniforme de recursos) de la petición.
- La versión http.

Ejemplo: GET /miruta.html http/1.0

Después de la línea de petición llegan ***las Cabeceras de Petición:*** Se trata de pares clave/valor, separados por dos puntos (:) de modo que para cada par ocupa una línea. Las cabeceras de petición proporcionan al servidor información adicional sobre la identidad y las capacidades del cliente. Las cabeceras de petición más comunes son:

- User-Agent: La marca y la versión del cliente (o navegador Web)
- Accept: Una lista de tipos de contenido que el cliente reconoce
- Content-Length: El número de bytes de datos anexados a la petición.

La cabecera http que envía el cliente contiene los detalles acerca de los tipos de documentos que aceptará del servidor. La información de la cabecera se divide en tres partes: *General*, que contiene información sobre el cliente o servidor, que no es específica. *Entity*: contiene información de los datos enviados al servidor. *Request*: contiene información sobre la configuración del cliente.

En tercer lugar, tras la última cabecera de petición se envía en una ***línea en blanco*** que consiste únicamente en un retorno de carro. Esto informa al servidor de que ya no quedan más cabeceras. Incluso cuando no hay ninguna cabecera, se debe enviar esta línea en blanco para que el servidor no busque cabeceras.
Por último ***el cuerpo http*** del cliente contiene la información que envía al servidor, que pueden ser datos de un formulario que van en un mensaje de tipo post.

Respuesta HTTP del Servidor

La respuesta HTTP similar a la petición, consta de un máximo de cuatro partes: una línea de estado; ninguna, una o varias cabeceras de respuesta; una línea en blanco que señala el final de las cabeceras; y los datos que conforman la petición.

La línea de estado tiene hasta 3 categorías:

- *La versión HTTP:* Del mismo modo que el cliente indica la versión más avanzada que es capaz de entender, y contiene un código del estado al procesar la solicitud del cliente. Los códigos informan acerca del éxito de la solicitud, o errores en el cliente o servidor.
- *El código de respuestas:* Es un código numérico de tres dígitos que indica si la petición se ha realizado correctamente y, en cado de fallo, la razón por la que éste se ha producido.
- *Una descripción opcional de la respuesta:* Que es una explicación del código de respuesta comprensible para el usuario.

La cabecera http del mensaje de respuesta es similar a la del mensaje de petición. Esta dividida en tres tipos de petición: *General,* contiene información acerca del cliente o el servidor. *Entity:* contiene información acerca de los datos enviados entre el cliente y el servidor. *Response:* Contiene información acerca del servidor y como el cliente puede manejar la respuesta.

Tras la última cabecera de petición se envía en una ***línea en blanco*** que consiste únicamente en un retorno de carro. Esto informa al servidor de que ya no quedan más cabeceras.

El cuerpo http de la respuesta contiene el código HTML para que el browser del cliente lo interprete.

2.2.3.3 Servlets

Los Servlets son clases Java que amplian la funcionalidad de un servidor Web mediante la generación dinámica de páginas Web. Un entorno de ejecución denominado *motor de servlets* administra la carga y la descarga del servlet, y trabaja con el servidor Web para dirigir peticiones a los servlets y enviar la respuesta a los clientes.

Los Servlets aportan varias ventajas clave:

- Rendimiento: Las tecnilogías anteriores como la Interfaz de pasarela CGI (Common Gateway Inerface) normalmente inician un nuevo proceso para manejar cada petición que les llega. Los servlets, al contrario, se cargan cuando los solicitamos por primera vez y permanecen indefinidamente en la memoria. El motor de servlet carga un solo ejemplar o instancia de la clase Servlets y le lanza peticiones empleando un conjunto de subprocesos disponibles (threads o hilos).
- Simplicidad: Los applets Java del cliente se ejecutan en una máquina virtual proporcionada por el navegador Web. Esto genera problemas de compatibilidad que limitan la funcionalidad de los applets. Los Servlets simplifican esta situación ya que se ejecutan en una máquina virtual en un entorno de servidor controlado y sólo necesita el HTTP básico para comunicarse con sus clientes.
- Sesiones HTTP: Aunque los servidores HTTP no tienen capacidad para recordar detalles de una petición previa del mismo cliente, la interfaz API Servlet proporciona una clase **HTTPSession** que permite superar esta limitación.
- Acceso a la tecnología Java: Al ser aplicaciones Java, los Servlets tienen acceso directo a toda la gama de características Java, como el uso de subprocesos, acceso a redes y conectividad a bases de datos.

Las páginas JSP, que se convierten automáticamante en Servlets, heredan todas estas ventajas.

CICLO DE VIDA DE UN SERVLET

Los Servlets operan en el contexto de un modelo de petición y respuesta administrado por el motor de servlets. El motor de Servlets se encarga de:

- Carga un Servlet cuando lo solicitamos por primera vez.
- Llama al método **init( )** del servlet.
- Manejar todas las peticiones que reciba llamando al método del servlet **service( )**.
- Manejar al método **destroy( )** de cada servlet al terminar la ejecución.

API SERVLET

La interfaz API Servlet 2.3 contiene un conjuto de clases divididas en dos paquetes servlet:

- Javax.servlet : Clases servlet no específicas de ningún protocolo
- Javax.servlet.http: Clases servlet específicas del protocolo HTTP.

Interfaces de la API Servlet (*Paquete javax.servlet*)

A continuación se presenta un subconjunto de éstas, que son de uso más frecuente:

Servlet : Define los métodos que todos los servlets deben implementar. La API de servlet proporciona implementaciones concretas de esta interfaz en las clases *GenericServlet* y *HttpServlet***.**

ServletConfig: Utilizado por el motor de servlets, usado para pasar información a un servlet durante la inicialización

ServletContext: Define la lista de métodos que están disponibles para que un servlet se comunique con su contenedor de servlet. El contexto del servlet puede

almacenar atributos que quedan disponibles para que todos los servlets de la aplicación.

ServletRequest: Interfaz que representa una petición de un cliente a un servlet. El motor de servlets crea un objeto *ServletRequest* y lo pasa como argumento al método *service* del servlet. Tiene métodos para recuperar nombres de parámetros, valores atributos y el flujo de entrada.

ServletResponse: Encapsula toda la información acerca de la respuesta generada para una petición, incluyendo las cabeceras de respuesta, el código de estado y el flujo de salida. *HttpServletResponse* amplía esta interfaz para características http específicas.

Existen otras interfaces como:
Java.servlet.Filter
RequestDispatcher
ServletContextAttributesListener
ServletContextListener
SingleThreadModel

Clases Escenciales de API Servlet (*Paquete javax.servlet*)

GenericServlet: Clase abstracta base para todos los servlets que no utilizan características específicas del protocolo http. *GenericServlet* implementa las características básicas de todos los servlets: Inicialización, gestión de peticiones y terminación.

ServletContextAttributeEvent : Utilizada para notificaciones sobre cambios en los atributos del contexto del servlet de una aplicación Web.

ServletContextEvent: Clase de evento para notificaciones sobre cambios en el contexto del servlet

ServletException: maneja una excepción de servlet genérica.

ServletResponseWrapper: Clase base para subclases que implementan *ServletResponse*. Su comportamiento predeterminado es invocar los métodos correspondientes de la clase de implementación *ServletResponse* del motor de servlets.

UnaVailableException: Subclase de *ServletException* lanzada por un servlet cuando ya no se pueden manejar peticiones, de forma provisional o permanente.

Interfaces de la API Servlet (*Paquete javax.servlet.http*)

HttpServletRequest: Encierra toda la información sobre una petición http: los parámetros, los atributos, las cabeceras y los datos de entrada.

httpServletResponse: Encapsula toda la información sobre la respuesta generada por una petición HTTP, incluyendo las cabeceras de respuesta, el código de estado y el flujo de salida.

HttpSession: Es un repositorio o almacén de referencias con nombre a objetos pertenecientes a la sesión del navegador de un usuario. Este repositorio permanece activo en el servidor entre las peticiones del usuario. Una sesión tiene un ID de sesión único asignado por el servidor del cual el cliente deja una traza y lo pasa de vuelta con cada petición subsiguiente.

HttpSessionContext: Anteriormente utilizado para permitir la comunicación directa entre servlets, esta clase de desaconseja actualmente.

HttpSessionListener: Las clases que implementa esta interfaz, reciben notificación cuando se crean o se invalidan sesiones.

Existen otras interfaces como:

HttpSessionActivationListener
HttpSessionAttributesListener
HttpSessionBindingListener

Clases escenciales de api servlet (*Paquete javax.servlet.http*)

HttpServlet: Clase base abstracta para servlets que operan en un entorno http. HttpServlet es una extensión ligera de GenericServlet que proporciona métodos específicos para http para peticiones Get, Post, Put, Delete, Head, Options y Trace. El método service( ) determina el tipo de petición http e invoca al método apropiado.

HttpSerrvletRequestWrapper: Esta clase es una implementación concreta de HttpServletRequest, que puede ser sobreescrita de manera neutra con respecto al motor de servlets para proporcionar funcionalidad adicional al objeto de la petición. Por defecto, comprueba los métodos correspondientes específicos de motor de servlets.

HttpSerrvletResponseWrapper: Implementación concreta de httpServletResponse que puede extenderse para permitir la personalización del objeto de respuesta. Por defecto, los métodos de esta clase comprueban a sus homólogos de la clase de implementación del motor de servlets.

HttpSessionEvent: Representa una notificación de eventos para cambios en las sesiones de una aplicación Web.

2.2.3.4 Páginas Jsp

JSP es la tecnología desarrollada por Java para la creación fácil de contenidos dinámicos dentro de documentos html en el lado del servidor. JSP permite separar la forma como se presenta el contenido del documento o página web, de la lógica usada (llamada también lógica del negocio) para generarlo. Esto trae como gran ventaja la posibilidad de modificar la presentación sin tener que modificar la lógica, o viceversa.

Básicamente JSP es simplemente una página HTML, que contiene partes adicionales de código del lenguaje de programación JAVA, el cual es ejecutado cuando se hace una solicitud al servidor y este, en respuesta genera contenido HTML dinámico. Las páginas JSP son una extensión de la tecnología Servlet de Java, que también permite crear contenido web de forma dinámica, pero en la cual, el código html está embebido dentro de las clases de JAVA. Esto trae como

complicación que un cambio que se tenga que hacer en la presentación implica la modificación de una clase Java y por lo tanto su recompilación. Esto se soluciona con JSP, ya que es una forma de Servlets de alto nivel en los cuales el código de Java es el embebido en la página HTML. Como son Servlets, las páginas JSP tienen todas las ventajas de los Servlets:

- Tienen un mayor rendimiento y capacidad de adaptación (llamado escalabilidad) que las secuencias de comandos CGI, porque se conservan en la memoria y admiten múltiples subprocesos.
- No es necesaria una configuración especial por parte del cliente.
- Incorporan soporte para sesiones HTTP, lo que hace posible la programación de aplicaciones.
- Tienen pleno acceso a la tecnología Java (capacidad de reconocimiento del trabajo en red, subprocesos y conectividad a bases de datos) sin las limitaciones de los applets del cliente.

JSP soporta todas las características de JAVA como el uso de clases, lo que permite la modularidad del software, en el cual la página JSP contiene la presentación de la información, y hace los llamados a clases que contienen la lógica del negocio.
Pero además las páginas JSP tienen ventajas propias:

- Se vuelven a compilar automáticamente cuando es necesario.
- Como están en el espacio común de documentos del servidor Web, dirigirse a ellas es más fácil que dirigirse a los Servlets.
- Como las páginas JSP son similares al HTML, tienen mayor compatibilidad con las herramientas de desarrollo Web.

Una página JSP consta de los siguientes elementos:

Código HTML: Corresponde a la mayoría del contenido de la página y es la manera como está formateado el documento para que el navegador del cliente lo interprete.

Directivas: Afectan a la estructura general de la clase servlet de la página JSP. Son instrucciones dirigidas al contenedor de JSP que describen el código que se debe generar. La forma de uso es la siguiente:

<%@ nombre_directiva nombre_atributo="valor" %>

Las directivas más importantes son: *page* que permite definir las clases o paquetes de clases que la página usará, el tipo de contenido, el uso de la sesión, etc. *Include* que permite incluir archivos en el documento.

Expresiones: Son usadas para insertar valores JAVA directamente en la página. En otras palabras es un modo sencillo para obtener acceso al valor de una variable Java u otra expresión y unir ese valor con el HTML de la página. Se escribe de la siguiente manera:

<%=Expresión de Java%>

Scriplets: Son los elementos que permiten insertar código Java de forma arbitraria dentro de las páginas. En otras palabras un Scriplet es un conjunto de una o más sentencias o instrucciones (statements) en lenguaje Java concebidas para su uso en el procesamiento de una petición HTTP. Su código se convierte en parte del método -jspService(). Se escriben de la siguiente manera:

<%Código de Java %>

Declaraciones: Permiten definir métodos. Al igual que los scriplets, las declaraciones contienen instrucciones (statements) con la diferencia que su código se incorpora en el fichero fuente generado, fuera del método _jspService(). Se escriben de la siguiente manera:

<%! Código de Java %>

Comentarios: Existen dos formas de incluir comentarios en una p{agina JSP: una para comentarios ocultos sólo visibles en la propia página JSP y otra para

comentarios incluidos en la salida HTML generada por la página. Se escriben de la siguiente manera:

<%-- Comentario JSP oculto --%>
<!-- Comentario que incluye el HTML generado -->

CICLO DE VIDA DE UNA PÁGINA JSP

Comienza cuando el programador escribe la página JSP y es cargada en un servidor web que está habilitado para manejar JAVA. La primera vez que la página JSP es solicitada por un cliente, esta es compilada en una clase de JAVA que corresponde a las instrucciones de la página JSP. Esto quiere decir que la página no es interpretada directamente por el servidor, sino que debe compilarla primero para poder ejecutarla. La página es compilada en un Servlet (una clase de java que es ejecutada en el servidor), el cual recibe las instrucciones del cliente, las procesa y le entrega respuestas. Los Servlets quedan residentes en memoria, por lo que son compilados únicamente cuando son solicitados la primera vez.

API JSP

La interfaz API JSP 1.2 contiene un conjuto de clases divididas en dos paquetes de JSP:

- Javax.servlet.jspBase: Clases base de páginas Java en servidor (JavaServer Pages).
- Javax.servlet.jso.tagext: Etiquetas personalizadas JSP.

Interfaces de la api jsp (*Paquete javax.servlet.jspBase*)

A continuación se presenta un subconjunto de éstas, que son de uso más frecuente:

HttpJspPage: Es Una subinterfaz de JspPage, implementada por clases específicas de HTTP generadas por un motor de JSP. El motor de JSP creará

automáticamante un método –jspService( ) que contiene todo el código del scriptlet definido en la página.

JspPage: Es una subinterfaz de Servlet, implemantada por clases generadas por un motor de JSP. Los métodos jspInit( ) y jspDestroy( ) pueden ser sobreescritos por el autor de JSP para llever a cabo lo que hacen los métodos ServletInit( ) y destroy( ).

Clases escenciales de api servlet (*Paquete javax.servlet. jspBase*)

JspException: Es la clase base genérica para excepciones de JSP. Esta excepción la lanza una serie de métodos que se encuentran en las clases de etiquetas personalizadas.

JspTagException: Subinterfaz de JspException utilizada en manejadores de etiquetas para indicar un error fatal.

JspWriter: Subclase de java.io.Writer que se utiliza para escribir salida JSP. Esta clase es instanciada por el –jspService( ) generado, invocando el método getWriter( ) del servlet subyacente, que hace ilegal llamar luego a getOutputStream.

PageContext: Objeto envoltorio que encpsula todos los detalles de una llamada simple de una JSP para gestionar una petición. Contiene métodos para inicializar y lanzar los objetos sesión, escritor, petición y respuesta.

Existen otras clases como:
JspEngineInfo
JspFactory

Interfaces de la api jsp (*Paquete javax. servlet.jsp.tagext*)

Tag: Es un conjunto de métodos de ciclo de vida que deben implementar los manejadores de etiquetas personalizadas.

IterationTag: Extensión de la interfaz Tag que define la semántica para la evaluación repetida del cuerpo de la etiqueta.

BodyTag: Extensión de la interfaz *IterationTag* que añade nuevos m{etodos relacionados con la gestión del cuerpo.

TryCatchFinally: Interfaz adicional que puede ser implementada por manejadores de etiquetas para permitirles que sean invocados en los bloques *catch* y *finally* de la llamada de etiquetas

Clases escenciales de api servlet (*Paquete javax. servlet.jsp.tagext*)

TagAttributeInfo: Clase que da información acerca de los atributos de una etiqueta, disponible en tiempo de traducción.

TagSupport: Implementación concreta de la interfaz Tag. Los manejadores de etiquetas pueden extender esta clase e implementar solamente los métodos que necesiten cambiarse.

Existen otras clases como:
BodyContent
BodyTagSupport
PageData
TagData
TagExtraInfo
TagInfo
TagLibraryInfo
TagLibraryValidator
TagVariableInfo
VariableInfo

2.2.4 Acceso a Bases de Datos

La conexión a bases de datos es necesaria para almacenar la información de catálogos de productos que tendrá el sitio Web, que luego será consultada por los clientes.

Una de las características principales de JAVA es su facilidad para conectarse a almacenes de datos externos como bases de datos relacionales. Para stablecer una conexión y la comunicación con la base de datos se usa el API JDBC o JAVA Database Conectivity. Esta API posee como principal característica que es única para todas las bases de datos.

2.2.4.1 Visión General de JDBC

JDBC es una interfaz de programación de aplicaciones ente los programas Java y los sistemas de administración de bases de datos. JDBC es una una interfaz a nivel de llamada, esto significa que un programa emplea llamadas a funciones o métodos para obtener acceso a sus características; en oposición a las instrucciones SQL incrustadas, traducidas por un precompilador.

Un programador emplea una clase Java conocida como *controlador JDBC* (JDBC driver) para conectarse a una base de datos. La gran ventaja de JDBC es que proporciona una interfaz estándar para todos los sistemas de administración de bases de datos. JDBC también facilita la transición de sistemas heredados a aplicaciones adaptadas a la red. Los protocolos SQL incrustados, que se han venido utilizando desde principios de los ochenta, emplean en su mayor parte instrucciones y operaciones SQL que se pueden duplicar mediante llamadas JDBC.

OPERACIONES BÁSICAS CON JDBC

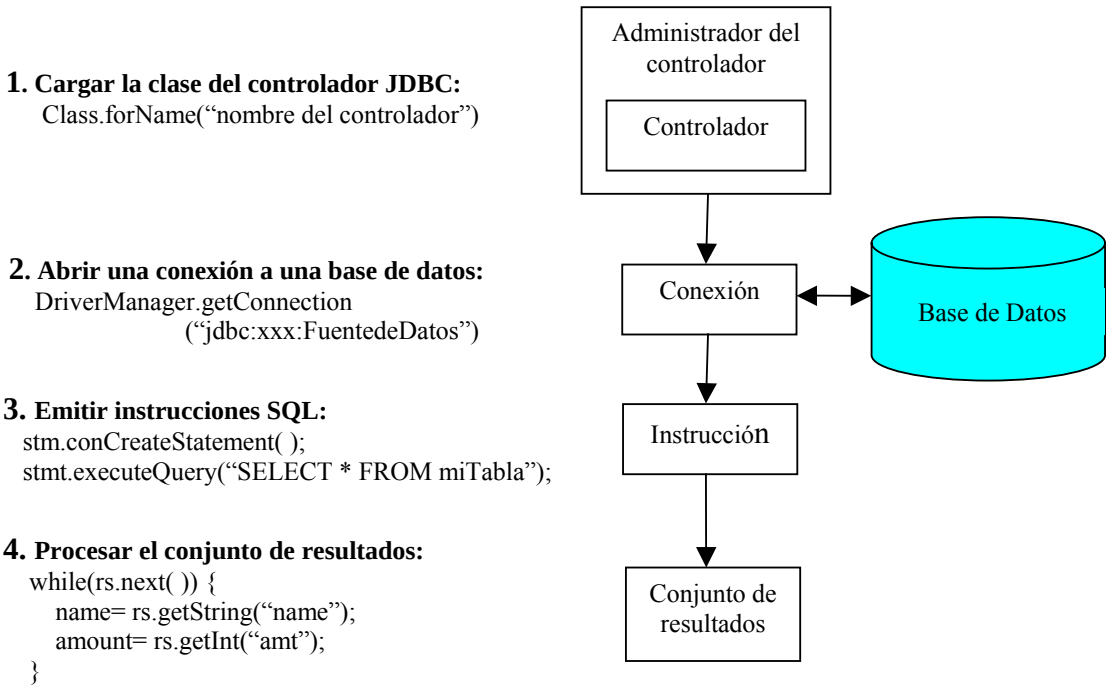
Dependiendo de la tarea a realizar, normalmente se deben seguir sólo 4 pasos:

- Cargar un controlador JDBC para su DBMS (Sistema de administración de bases de datos). Normalmente esto sólo requiere una instrucción **Class.forName()** que especifique el nombre de la clase del controlador.
- Utilizar ese controlador para abrir una conexión con una base de datos concreta. Esto se consigue con una llamada al método estático

getConection (url) de la clase **DriverManager.** El argumento url está en un formulario específico que indica el tipo de controlador y la fuente de datos que se va a emplear.

- Emitir instrucciones SQL a través de la conexión. Una vez establecida la conexión, está se puede emplear para crear objetos **Statement** mediante los cuales de puede realizar comandos SQL.
- Procesar los conjuntos de esultados devueltos por las operaciones SQL. La interfaz **ResultSet** proporciona métodos para pasar por cada una de las filas y obtener los valores de todas las columnas. La siguiente figura ilustra estos cuatro pasos.

Figura 3. Los cuatro pasos de las operaciones JDBC básicas



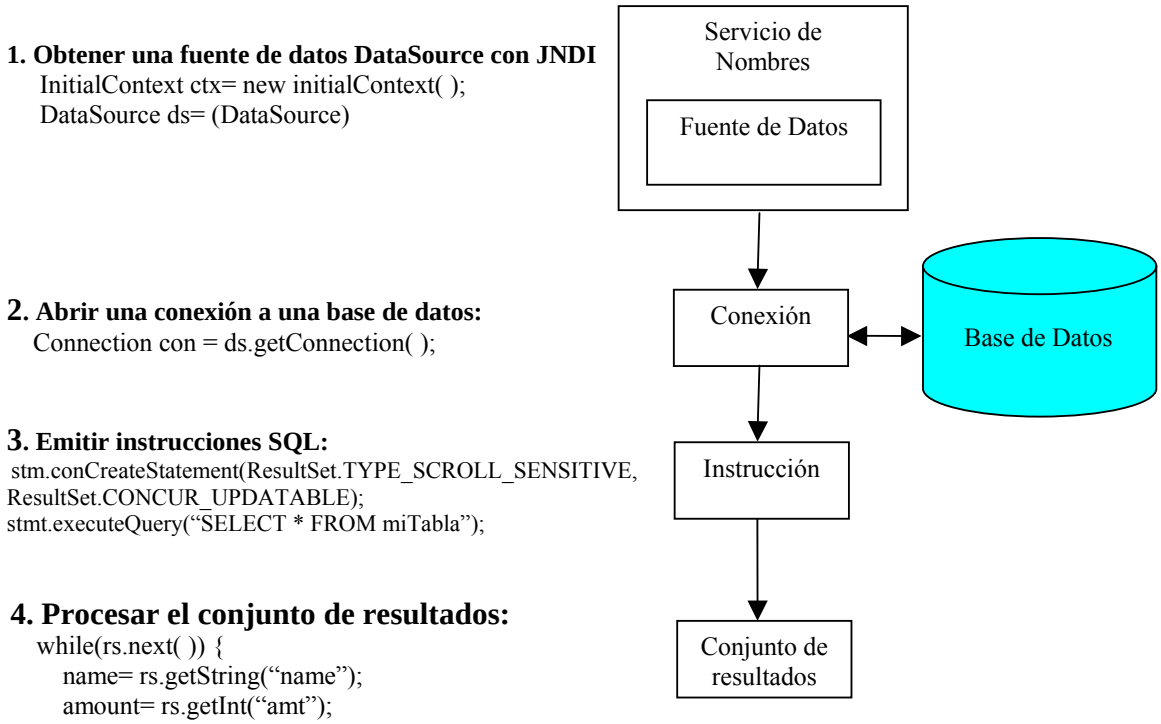
Desde JDBC 2.0 en adelante, hay algo más de la flexibilidad. Empleando *Java Naming and Directory Interface* (JNDI, Interfaz de servicios de nombres y directorios Java), una aplicación puede buscar un objeto **DataSource** por su nombre en un sevicio de nombres y no tiene que incluir en el programa el

nombre de clase del controlador y el URL de la base de datos. Además, los conjuntos de resultados **JDBC** (2.0 en adelante), tienen más capacidades.

Se pueden obtener acceso a ellos de ofrma aleatoria u no necesariamente de forma secuencial desde el principio hasta el final. Se pueden actualizar y transmitir las actualizaciones a la tabla subyacente de origen. También se pueden vincular dinámicamente a su tabla o tablas base, de modo que los cambios reflejen simultáneamente en el conjunto de resultados.

La siguiente figura muestra los pasos básicos de acceso a base de datos con JDBC 2.0 en adelante.

Figura 4. Acceso a Bases de Datos con JDBC y JNDI



Interfaces de la api de jdbc

Java proporciona distintas interfaces para conectarse a la base de datos y ejecutar instrucciones de SQL. Mediante las interfases de la API (interfaz de programación de aplicaciones) de JDBC (Java Database Connectiviry) usted puede ejecutar instrucciones normales de SQL, instrucciones dinámicas de SQL y procedimientos almacenados que emplean los parámetros IN y UOT.

La interfaz de JDBC se encuentra en los paquetes **java.sql** y **javax.sql.** Consiste principalmente en interfaces más que en clases concretas porque la implementación de cada empresa distribuidora, es específica del protocolo de bases de datos concreto.
El núcleo de la API de **java.sql** consta de 16 interfaces, 8 clases y 4 tipo de excepciones. La API Optional Package añade otras 12 interfaces y 2 clases.

A continuación se presenta un subconjunto de éstas que son de uso más frecuente:

CallableStatement: Le permite ejecutar procedimientos almacenados que retornan valores del parámetro OUT. El objeto CallableStatement, hereda del objeto PreparedStatement, pero también agrega varios métodos para registrar parámetros que sean OUT.

Connection: Es el objeto que proporciona a una aplicación JAVA una conexión a la base de datos. Este objeto se emplea para crear todos los distintos objetos Statement para ejecutar instrucciones de SQL y procedimientos almacenados.

DatabaseMetadata: Proporciona diversos métodos para obtener información sobre la base de datos. Proporciona métodos para obtener el listado de las tablas de una base de datos, así como las claves primarias, las columnas, etc.

Driver: El objeto driver, es una objeto para una base de datos específica proporcionado por el distribuidor de JDBC. Contiene información específica sobre la conexión de su aplicación JAVA.

PreparedStatement: Le permite ejecutar instrucciones dinámicas de SQL y procedimientos almacenados.

ResultSet: Es el objeto que se crea y se utiliza para obtener información de una instrucción Select de SQL. Esta instrucción retorna un cursor que es utilizado por la interfaz ResultSet par navegar a través de los resultados retornados por la instrucción Select.

ResultSetMetaData: Le permite obtener información sobre un conjunto de resultados retornado. El objeto ResultSetMetaData se crea a partir de un objeto ResultSet y proporciona información específica a ese objeto.

Statement: Esta interfaz se crea a partir del objeto Connection y se puede usar para ejecutar instrucciones SQL estándar y procedimientos almacenados. El objeto proporcina dos métodos principales: executeQuery( ) y executeUpdate( ), que permiten ejecutar consultas SQL y actualizaciones de SQL.

Clases esenciales de jdbc

Java también proporciona un puñado de objetos que se pueden usar en las aplicaciones Java. La mayoría de objetos se emplean para proporcionar a Java algunos de los tipos de datos específicos para base de datos disponibles en la mayoría de las bases de datos.

Date: Proporciona un objeto que puede aceptar valores de tipo Date de base de datos. Este objeto es heredado del objeto Date normal de Java.

DriverManager: Proporciona otra forma de realizar una conexión a la base de datos. El objeto se usa principalmente para administrar objetos Driver de JDBC. Proporciona varios métodos para el registro de controladores, obtención de conexiones y envío de información al flujo de salida de la base de datos.

DriverPropertyInfo: Es utilizado por programadores avanzados para manejar propiedades específicas de un objeto Driver.

Time: Es heredado del objeto Date de Java. Proporciona un objeto que puede aceptar valores de tipo timestamp de base de datos y varios métodos para obtener y asignar valores del objeto.

Timestamp: Puede utilizarse para obtener valores de datos de la base de datos que son del tipo Timestamp. El objeto proporciona métodos para comparar los valores de dos diferentes objetos Timestamp.

Types: Contiene una lista de valores enteros predefinidos que identifican cada uno de los diferentes tipos de datos disponiobles para uso en aplicaciones JDBC.

Excepciones de jdbc

Cuando ocurre un error en Java, se lanza una excepción. Cualquier excepción que lancen los métodos de Java debe ser "atrapada" por el usuario. La API de JDBC contiene tres excepciones que pueden atraparse identificando varios errores en la ejecución de métodos JDBC. A continuación se enumeran:

DataTruncation: Se lanza siempre que JDBC trunca un valor de datos en forma inesperada. La excepción proporciona métodos para obtener información sobre el valor de datos que fue truncado así como para obtener información sobre el error de trucamiento.

SQLException: Es lanzada por casi todos los métodos en la API de JDBC. Esta exception proporciona varios métodos para obtener información sobre el error y el estado actual de la transacción SQL.

SQLWarning: Se genera cuando la base de datos emite una advertencia. La advertencia se envía silenciosamente al objeto que la causó.

CONTROLADORES JDBC

Para aislar los programas de los protocolos específicos para bases de datos concretas, JDBC emplea una capa intermedia compuesta de una clase **DriverManager** y uno o más controladores JDBC.

Un controlador es una clase Java, normalmente proporcionada por la empresa distribuidora de la base de datos, que implementa la interfaz **java.sql.Driver.**
La función principal del controlador, es conectarse a una base de datos y devolver un objeto **java.sql.Connection.**

Los programas de aplicaciones no pueden llamar directamente a los controladores. En vez de esto, son reistrados por **DriverManager**, que determina el controlador apropiado para una petición de conexión concreta y realiza la conexión a través de ese controlador.

Tipos de controladores

La especificación de JDBC clasifica los controladores en cuatro tipos, según su estructura. Estos tipos son:

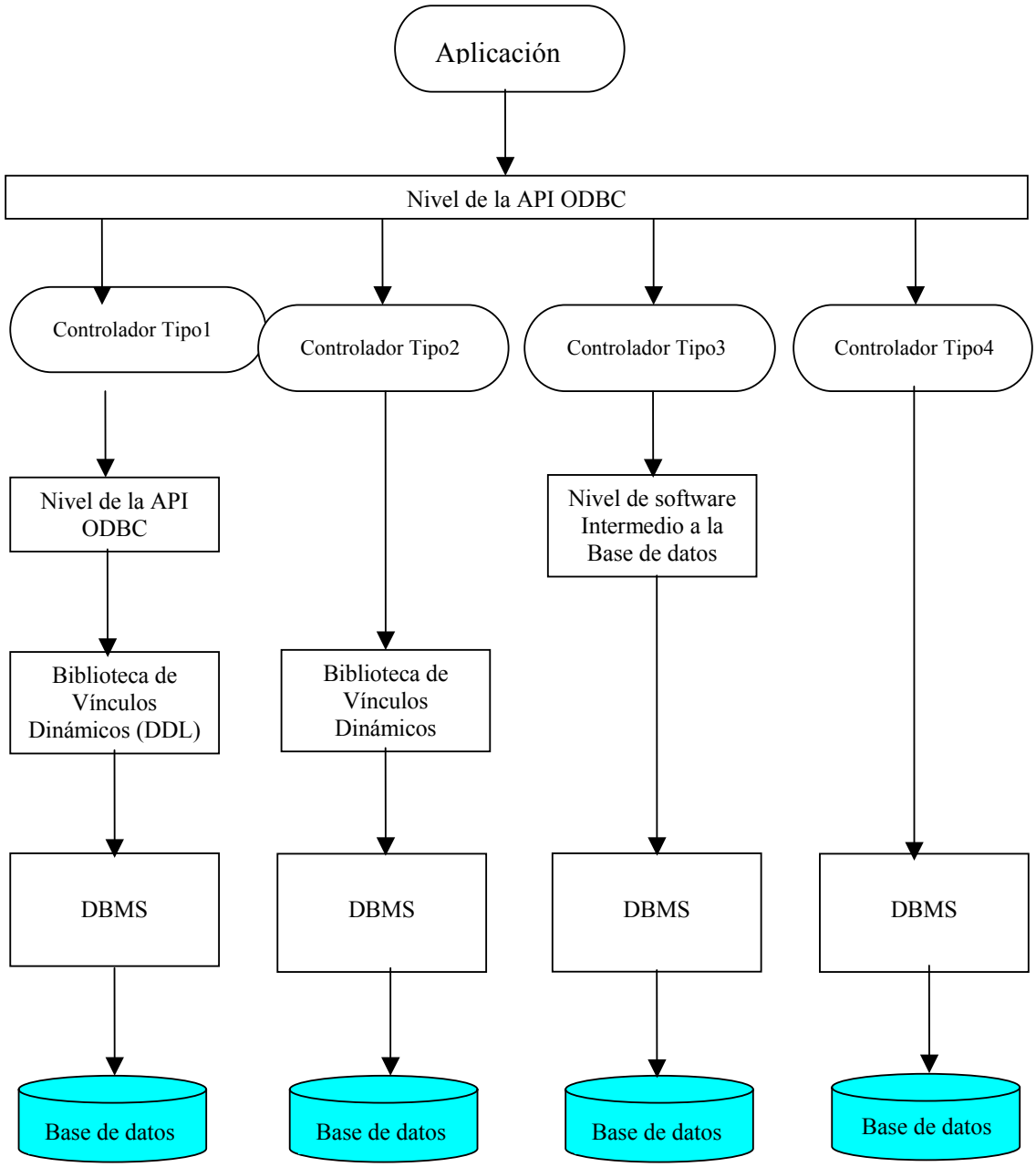
- **Tipo1 – Puente JDBC-ODBC:** Los controladores de este tipo se conectan a bases de datos mediante un controlador ODBC intermedio. Este sistema tiene varios inconvenientes, así que Sun lo describe como experimental y adecuado únicamente cuando no hay otro controlador disponible. Tanto Microsoft como Sun proporcionan estos controladores.

- **Tipo2 – API nativa, parcialmente en Java:** Similar al puente JDBC-ODBC, este controlador emplea métodos nativos para llamar a las funciones API específicas de la empresa distribuidora. Estos controladores están sometidos a las mismas limitaciones que el puente JDBC_ODBC, ya que necesitan que los ficheros de las bibliotecas nativas estén instalados en los sistemas de los clientes, que deben estar configurados para poder utilizarlos.

- **Tipo3 – Java puro con software intermedio(middleware) a base de datos:** Estos controladores establecen comunicación mediante un protocolo de red a un servidor de software intermedio, que a su vez establecen contacto con uno o más sistemas de administración de bases de datos.

- **Tipo 4 – Java puro directo a bases de datos:** Los controladores de este tipo llaman directamente al protocolo nativo empleado por el sistema de administración de la base de datos.

Figura 5. Las estructuras de los cuatro tipos de controlador JDBC



JDBC proporciona dos objetos diferentes para el manejo de controladores de bases de datos. Estos dos métodos son:

La Interfaz Driver: Ofrece varios métodos para obtener información sobre el controlador actual de la base de datos. También proporciona el método connect( ), el cual crea un objeto connection que puede utilizarse para accesar la base de datos.

El Objeto Connection: Es el principal objeto utilizado para proporcionar un vínculo entre la base de datos y sus aplicaciones Java. Este objeto le permite crear todos los objetos Statement a usar para la ejecución de instrucciones SQL y para obtener resultados de la base de datos. Puede emplear tanto el objeto Driver como el DriverManager para crear un objeto connection. Puede usar el método connect( ) y getConnection( ) respectivamente con el fin de crear un objeto connection para su uso en sus aplicaciones.

2.2.5 Servidores Web

Los Servidores Web suministran páginas web a los navegadores (como por ejemplo, Netscape Navigator, Internet Explorer de Microsoft) que lo solicitan. En términos más técnicos, los servidores Web soportan el Protocolo de Transferencia de Hypertexto conocido como HTTP (HyperText Transfer Protocol), el estándar de Internet para comunicaciones web. Usando HTTP, un servidor web envía páginas web en HTML y CGI, así como otros tipos de scripts a los navegadores o browsers cuando éstos lo requieren.

Cuando un usuario hace click sobre un enlace (link) a una página web, se envía una solicitud al servidor Web para localizar los datos nombrados por ese enlace. El servidor Web recibe esta solicitud y suministra los datos que le han sido solicitados (una página HTML, un script interactivo, una página web generada dinámicamente desde un base de datos,...), o bien devuelve un mensaje de error. Apache, el servidor Web suministrado con este producto, es el más usado en Internet actualmente.

2.2.5.1 Servidor Web Apache

Apache surgió a partir del servidor de HTTP más famoso y difundido en su época: NCSA. Desde entonces se convirtió en un poderoso rival de todos los servidores Unix utilizados hasta la fecha por su eficiencia, funcionalidad y rapidez. Es por ello que se conoce como el rey de los servidores *Web*. Se desarrolla de forma estable y segura gracias a la cooperación y los esfuerzos de un grupo de personas conocidas como grupo Apache (*Apache Group*), los cuales se comunican a través de Internet y del *Web*. Juntos se dedican a perfeccionar el servidor y su documentación regidos por la ASF (*Apache Software Foundation*).

En la actualidad Apache es el servidor *Web* más utilizado en el mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. De acuerdo con las estadísticas, lo colocan en más de 7 millones de servidores que sirven poco más de 18 millones de sitios *Web*, lo cual significa más del 60% en todo el mundo.

Es un software de código abierto que funciona sobre cualquier plataforma. Por supuesto, se distribuye prácticamente con todas las implementaciones de Linux. Tiene capacidad para servir páginas tanto de contenido estático, para lo que nos serviría sencillamente un viejo ordenador 486, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información.
Entre las características principales del Apache se encuentran:

- Es un servidor *Web* potente, flexible y ajustado al HTTP/1.1
- Es altamente configurable y extensible.
- Puede ser ajustado a través de la definición de módulos empleando su propio API (*Aplication Programming Interface*).
- Provee todo su código fuente de forma libre y se distribuye bajo una licencia no restrictiva.
- Se ejecuta en diversas plataformas operativas tales como: Windows 9x/NT, Macintosh, Novell NetWare, OS/2, Linux y la mayoría de los Unix existentes: IRIX, Solaris, HPUX, SCO, FreeBSD, NetBSD, AIX, Digital Unix, etc.

- Se desarrolla de forma acelerada estimulando la retroalimentación desde sus usuarios a través de nuevas ideas, reportes de errores y parches.
- Apache significa ``*A PAtCHy sErver*'', o sea se basa en un código y un conjunto de ficheros ``parches''. Otros desarrolladores relacionan su nombre con el de las tribus nativas americanas de Apaches.
- Implementa muchas posibilidades frecuentemente demandadas, tales como:

Bases de datos DBM para autenticación: Permiten establecer fácilmente la protección de documentos a través de *passwords* para una gran cantidad de usuarios sin dañar el funcionamiento del servidor.

Respuestas adaptables a los errores o problemas: Se pueden definir ficheros o *scripts* de tipo CGI, que respondan ante la ocurrencia de errores internos o en las solicitudes realizadas.

Directiva para definir múltiples índices: Se utiliza cuando se solicitan directorios por parte de los clientes a partir de lo cual se puede buscar en estos y devolver un documento índice cuyo nombre puede ser por ejemplo: `index.html`, `index.cgi` o `default.html`.

Ilimitadas y flexibles posibilidades de redireccionamiento y definición de alias para los URLs: Apache no tiene un límite establecido para definir alias y redireccionamientos que pueden ser declarados en sus ficheros de configuración.
Negociación del contenido de las respuestas: Apache es capaz de ofrecer la mejor representación de la información accedida de acuerdo con las capacidades del cliente solicitante.

Soporte de hosts virtuales: Es la habilidad del servidor de distinguir entre los pedidos hechos a diferentes direcciones IP o nombres de dominio definidos en la misma máquina.

Configuración flexible de las trazas generadas: Es posible adaptar el formato de las trazas obtenidas así como redireccionarlas a través de tuberías (Unix) en aras de filtrarlas. De esta forma se puede lograr por ejemplo dividir dinámicamente las trazas de los *hosts* virtuales en distintos ficheros.

El servidor apache como servidor web seguro

El servidor Web Apache está diseñado de forma modular; consiste en muchas porciones de código que hacen referencia a diferentes aspectos o funcionalidades del servidor Web. Esta modularidad es intencionada, con lo cual, cada desarrollador puede escribir su propia porción de código para cubrir una necesidad en particular. Su código, llamado módulo, puede ser integrado en el servidor Web Apache con relativa facilidad.
El módulo mod_ssl es un módulo de seguridad para el Servidor Web Apache. El módulo mod_ssl usa las herramientas suministradas por el OpenSSL Project para añadir una característica muy importante al Apache, "la posibilidad de encriptar las comunicaciones". A diferencia de las comunicaciones entre un navegador y un servidor web usando HTTP "normal", en la que se envía el texto íntegro, pudiendo ser interceptado y leído a lo largo del camino entre servidor y navegador.

El OpenSSL Project incluye un kit de herramientas que implementa los protocolos SSL (Secure Sockets Layer) y TLS (Transport Layer Security), así como una librería de codificación de propósito general. El protocolo SSL se usa actualmente para la transmisión de datos segura sobre Internet; El protocolo TLS es un estándar de Internet para comunicaciones privadas (seguras) y fiables a través de Internet. Las herramientas OpenSSL son usadas por el módulo mod_ssl para aportar seguridad en las comunicaciones Web.

2.2.5.2 Servidor Web Tom-Cat

TomCat es un servidor de páginas web habilitado para procesar JAVA, creado por la Apache Software Fundation. TomCat recibe las solicitudes de los clientes por páginas html, JSP, o Servlets escritos en java, y envìa mensajes de regreso con las respuestas a las solicitudes a través del protocolo http.

TomCat es la implementación de referencia de las especificaciones JSP y Java Servlets, lo que significa que es el servidor que mejor cumple con el último estándar disponible de estas tecnologías.

La versión 4.0.1 de TomCat soporta el protocolo http 1.1 y las versiones 1.2 de JSP y 2.3 de servlets. El servidor está escrito completamente en JAVA, lo que garantiza su portabilidad a cualquier computador con la JVM instalada. Las características principales de TomCat son las siguientes:

Soporte de CGI: o Common Gateway Interfase, la cual define una forma en la cual un servidor web puede interactuar con los programas que generen contenido para el servidor.

Soporte de Aplicaciones Web: En TomCat, las páginas web pueden ser organizadas como aplicaciones. Cada aplicación tiene su propia estructura de páginas html, jsp o servlets, además de sus propias clases de JAVA. Esto permite el trabajo de varios proyectos simultáneamente en un mismo servidor, sin el temor a conflictos entre ellos, ya que cada uno es tratado de manera independiente.

Soporte de Seguridad: TomCat soporta el uso de la tecnología de seguridad con SSL en la cual el servidor se comunica con el cliente mediante el uso de certificados digitales que permiten la autenticación del cliente y el servidor, y la encriptación de los mensajes transmitidos. Esta característica permitirá que en el futuro, las transacciones que requieran un grado alto de seguridad puedan adoptar este mecanismo.

Multiple conexión: Soporta conexión de múltiples usuarios simultáneamente y manejo de sesiones para cada usuario.

El servidor TomCat es desarrollado bajo una licencia de GNU, en el cual los usuarios puedan usar el programa para cualquier propósito, acceder al código fuente y modificarlo para adaptarlo a sus necesidades, y publicar las mejoras realizadas para que todos se beneficien.

2.2.6 Manejadores de Bases de Datos

La información es un bien valioso, y las empresas no pueden descuidar su almacenamiento y manejo. Por eso, la tarea de resguardar los datos debe confiarse a un software que esté debidamente probado y que responda a las

necesidades sin falla alguna. De allí que la elección de los motores de bases de datos debe ser analizada con sumo cuidado.

Actualmente esta al alcance de sistemas operativos estandar para PC, la posibilidad de utilizar distintos motores de bases de datos, que brindan mayor seguridad en la información y posibilidades en lo que se refiere al manejo de los datos. Es factible utilizar Oracle, Informix, SQL Server, DB2 o Postgres, como motores de base de datos.

La arquitectura Cliente/Servidor es un enfoque para manejar aplicaciones de bases de datos con eficiencia, flexibilidad y control. Específicamente, el software Cliente/Servidor fracciona las aplicaciones monolíticas en componentes individuales, reusables y compatibles. Los clientes y servidores son independientes entre sí; no obstante, son totalmente interoperativos.
El componente Cliente maneja la interfase con el usuario y manipula los datos locales, mientras el componente Servidor se ocupa de los servicios relacionados con el manejo de los datos para diversos clientes. Los componentes Cliente y Servidor pueden correr en la misma computadora o en diferentes equipos, comunicados de una manera transparente en una red.

2.2.6.1 Motor de Base de Datos Oracle

Oracle es un sistema de administración de base de datos (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation. Se como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias). Otro aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma, y las políticas de suministro de

parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux.

3. ANALISIS Y DISEÑO

3.1 DISEÑO Y ARQUITECTURA GENERAL

A continuacion aparece las tres capas que forman parte de la arquitectura a seguir en el desarrollo de la aplicación web, esto nos permite desacoplar la lógica del negocio de la forma de presentación y de la persistencia, logrando de ésta forma, que cada capa esté especializada en una funcionalidad común, comunicándose, gracias a intancias de objetos Value Object que son repositorios de información para ser llevada entre una capa y otra. Las entidades que podemos ver en el diagrama son:

Paginas JSP:
Estos componentes se encargan de la presentación que tiene la aplicación para los diferentes actores, utiliza el framwork Struts 1.1 para el manejo de las diferentes peticiones web que llegan al sistema en desarrollo.

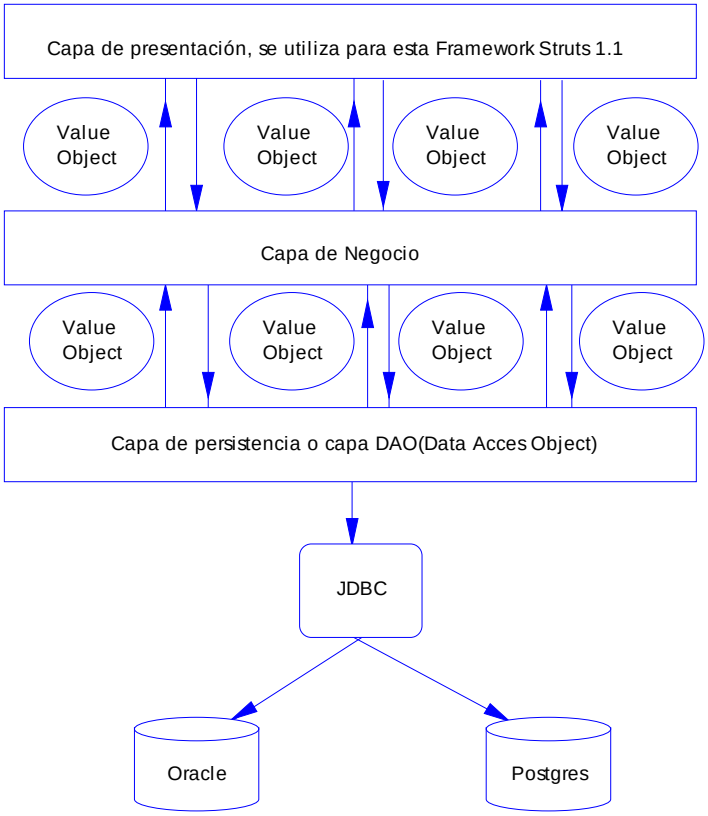
Value Object:
Esta clase facilita el envio de informacion entre la capa web y la capa del negocio, la capa del negocio y la capa de persistencia y por ultimo entre la capa de persistencia y la base de datos. Trabajan como repositorios de información permitiendo desacoplar las capas.

Bussines Object:
Objetos del negocio, estos componentes encapsulan la lógica del negocio, delegando la responsabilidad de la persintecia a la capa DAO y el despliegue a la capa de presentación.

Data Acces Object:

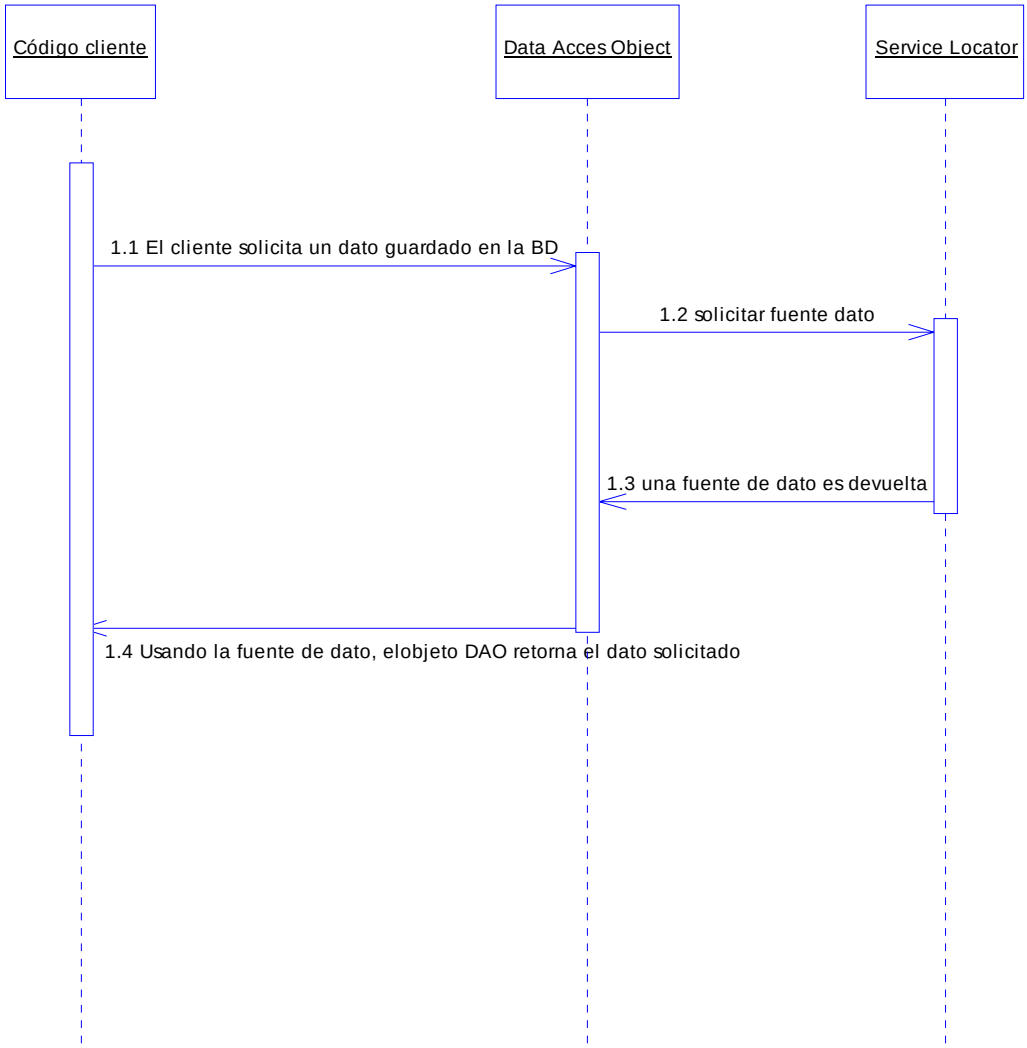
Estos componentes poseen la lógica para guardar los datos persistentes del negocio.

Figura 6. Capa de presentación



A continuacion aparece el diagrama de secuencia para la capa de persistencia:

Figura 7. Diagrama de secuencia para la capa de persistencia

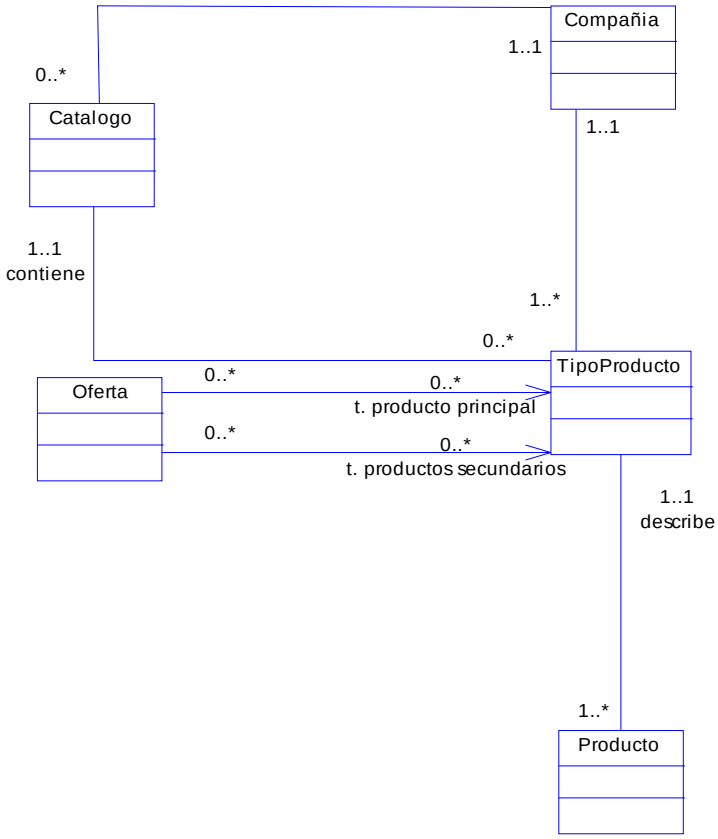


3.2 MODELO DE ANALISIS

Modelo de análisis

El primer paso en este proceso es escoger un modelo conceptual que describa los conceptos de éste dominio. No se hace referencia a cómo trabajará el software; únicamente el interes aquí, es saber como organizar los conceptos que hay en la mente de los comerciantes(vendedores) y compradores. A continuación aparece el modelo de dominio inicial para el sistema en desarrollo.

Figura 8. Modelo de dominio de ventas de productos



Las clases que aparecen se detallan a continuación:

TipoProducto:
Representa la descripción de un articulo que se puede vender en la tienda Pcware, no es el producto como tal, solo su definición; tiene atributos como nombre, precio, modelo, estado de inventario y descuento.

Producto:

Representa un artículo físico que se encuentra en inventario o que ya ha sido vendido a un cliente en la tienda Pcware, tiene una asociación con TipoProducto con multiplicidad uno a uno, lo que le permite a cualquier producto saber cual es su definición. En el otro sentido de esta relación podemos ver que muchos Productos pueden tener un misma descripción, lo cual esta de acuerdo con la realidad. Producto modela un objeto del mundo real, mientras que TipoProducto no tiene existencia sino sólo en el plano conceptual.

Oferta:
Esta abstracción es conceptual y nos permite hablar de las promociones como un solo concepto. Esta formada por un TipoProducto bajo el papel de tipo producto principal y una colección que desempeñan el rol de tipos productos secundarios.

Catalogo:
Esta clase facilita la organización de los diferentes Tipos de productos en categorías, lo cual nos facilita procesos de búsqueda al tratar un conjunto de tipo de productos como un todo bajo un sólo nombre.

Compañía:
Representa una entidad que produce y provee los diferentes productos vendidos por la empresa Pcware.

3.3 RIESGOS

Los riegos detectados hasta el momento en la capa de análisis y diseño son los siguientes:

Rendimiento inaceptable del sistema:
El uso del lenguaje de programacion Java como lenguaje de implementación puede llegar a ocasionar un sistema con bajo rendimiento y que esté tan lento, que se convierta en algo tedioso para los usuarios finales, pudiendo perderse ventas por parte de la empresa Pcware, esto es revelante sobre todo para aquellos casos de uso, con los que interactuan los actores clientes(compradores) del sistema.

Interfase de usuario no funcional:
La vista de despliegue de la aplicación es unicamente web, esto puede producir interfases de usuario poco amigables ya sea por la incompatibilidad existentes entre los diferentes navegadores web que hay en el mercado o por la utilización de componentes web no soportados o estádares, que al igual que el riesgo del numeral anterior pueda desencadenar pérdidas en las ventas de la compañía propietaria de la aplicación en el peor de los casos.

Poca capacidad del sistema para adaptarse a nuevos requerimientos:
El proyecto en estudio cubre sólo una parte de la lógica que envuelve un negocio de esta índole, el cual es bastante amplio y queda fuera del alcance de éste desarrollo. Su ejecución completa, por ejemplo, subsistemas como pagos en linea, soporte personalizado a usuarios, publicidad masiva, etc. son algunos de los puntos de extensión que se le pueden añadir al sistema una vez terminado, ésto implica que un precario desarrollo pueda originar una aplicación no robusta y no extensible.

Sistema no seguro:
La naturaleza web de la aplicación lleva asociada de forma implicita, un problema de seguridad, que de no ser definida a tiempo e impuesta en toda la arquitectura y en todos los niveles, puede ocurrir insatisfacción y no credibilidad en la tienda virtual por parte de sus usuarios, pricipalmente los clientes que son compradores potenciales de productos, ya sea de tipo clientes persona o cliente empresa.

3.4 TECNOLOGIA CANDIDATA

La Arquitectura a utilizarse para la aplicación, es de tres capas, basada en componentes J2SE Java 2 Standard Edition (usando además el API JSP 1.2, API Servlet 2.3, framework Struts 1.1, y Beans, para la aplicación Web); conformada por la capa de presentación (capa web), capa del negocio (capa media, capa BO) y la capa de persistencia de datos (capa DAO). Esta arquitectura, al final nos proporciona un marco de trabajo muy robusto basado en patrones(como se documenta en la parte teórica) de tal forma que se pueda solucionar problemas

en cuanto al rendimiento, escalabilidad y robustez del producto en desarrollo.(ver adelante en el Diseño, la arquitectura en lenguaje UML)

3.5 PRIMERA ITERACION

3.5.1 Fase de Análisis

En esta fase se ha realizado un trabajo ligero de requisitos para ayudar a decidir si merece la pena más investigación en el proyecto; se han definido la arquitectura a utilizar y una serie de riesgos (nombrados anteriormente) que podrían comprometer el desarrollo de la aplicación.

Inicialmente, se ha decidido abordar un escenario sencillo, del caso de uso Ver Ofertas, para sólamente un cliente cualquiera visitante(sin colaboraciones complicadas), con el objetivo de comenzar un diseño e implementación “amplio y superficial”, que incluya muchos de los elementos importantes del nuevo sistema.

3.5.1.1 Actores

Basados en los requerimientos funcionales del sistema se han identificados los actores cliente visitante, cliente registrado, administrador, director de ventas, auxiliar de soporte, auxiliar de atención al cliente, tecnico y auxiliar. En esta primera iteración sólo actúa el actor cliente

CLIENTE

Un cliente representa a una persona que es responsable de realizar cotizaciones, adquirir o comprar artículos (productos), solicitar un producto en particular o ver el estado en que se encuentra una orden de pedido ejecutada por él, como se describe en los casos de uso, realizar cotización, hacer pedido, solicitar producto y seguir pedido respectivamente.

Esta persona puede ser un individuo (cliente personal), es decir no asociado a una compañía, o alguien que representa una compañía (cliente compañía).

El cliente comprador de productos necesita el sistema tienda Pcware para enviar pedidos, solicitar productos particulares o realizar cotizaciones.

3.5.1.2 Casos de Uso

Durante esta fase se han identificado los siguientes casos de uso:

- Inicio visitante
- Ver oferta

DOCUMENTACION DEL CASO DE USO: INICIO VISITANTE – VER OFERTA

Nombre del caso de uso: Inicio visitante – Ver oferta

Actor principal: Cliente (visitante)

Historia de la revisión:

Fecha	Versión	Descripción	Autor
13/03/2005	1.0	Versión Inicial	Jeovany Enrique López Polo

Descripción:
El caso de uso inicio visitante permite a los clientes visitantes (visitantes en general) navegar por los servicios público (aquellos servicios de la aplicación que pueden ser accedidos sin ningún tipo de registro ni validación de usuario) de la tienda virtual, proporcionando a primera vista la oferta actual vigente, además del resto de los servicios ofrecidos por el sistema.

Precondiciones:
No hay una sesión de usuario creada.
Existe registrado por lo menos una oferta vigente con minimo 5 productos, de los cuales uno se considera como el producto principal de dicha oferta.

Restricciones de despliegue:

Los clientes visitantes pueden navegar por los servicios públicos de la aplicación (Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

Flujo de eventos (Sucesos):

- El usuario ingresa el URL de la aplicación y se le presenta las funcionalidades públicas de está, con la oferta actual vigente para un cliente visitante.
- El cliente invoca el caso de uso para iniciar como cliente visitante
- Se determina(asigna) que el cliente que solicita servicio a la aplicación es de tipo(rol) visitante.
- El sistema proporciona la oferta actual vigente para un cliente visitante, o sea la oferta principal.
- La aplicación encuentra la descripción general del producto principal asociado a la oferta.
- El sistema encuentra la descripción general de los productos secundarios asociados a la oferta.
- Se despliega en primera instancia la descripción general del producto principal asociado a la oferta.
- Se despliega de forma consecutiva la descripción general de los productos asociados a dicha oferta.
- El caso de uso termina

Flujo alterno de evento:

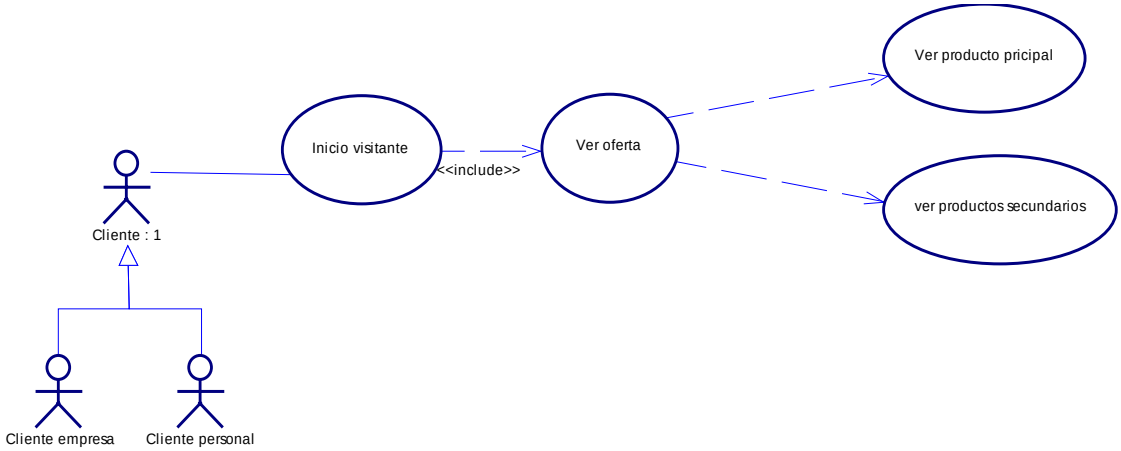
No hay oferta actual vigente para visitante:

- El sistema encuentra la descripción general de los cinco productos más recientes registrados al sistema
- El sistema ordena la descripción de los productos de forma descendente en base a la fecha de registro de la especificación del producto.
- Se despliega en primera instancia la descripción general del primer producto de la lista de los cinco encontrados en el numeral uno.
- Se despliega en forma consecutiva la descripción general del resto de los productos asociados a la lista.

Existe menos de cuatro especificaciones de productos secundarios para la oferta vigente:

- El sistema encuentra la descripción general de los cuatro productos más recientes registrados al sistema.
- El sistema ordena la descripción de los cuatro productos de forma descendente, en base a la fecha de registro de la especificación.
- El sistema completa la lista de los productos secundarios para la oferta, con la lista de los productos últimamente registrados.

Figura 9. Diagrama de casos de uso: Inicio Visitante – Ver oferta



3.6 SEGUNDA ITERACIÓN

3.6.1 Fase de Análisis

Después de reanalizar los requisitos exigidos por el sistema, y tomando como punto de partida la iteración anterior, se hace un incremento en el diseño del sistema, agregando otros casos de uso que ayuden a complementar el proyecto.

3.6.1.1 Actores

Basados en los requerimientos funcionales del sistema se han identificado otros actores aparte del cliente, como el administrador, director de ventas, auxiliar de soporte, auxiliar de atención al cliente, técnico y auxiliar. En esta segunda iteración se involucran todos los actores

DIRECTOR VENTAS

Un director de ventas es un empleado de la empresa propietaria del sistema Pcware encargada del control de artículos (tipo de productos y productos), procesar envíos pendientes y el envío de ofertas especiales para clientes registrados (promociones).

ADMINISTRADOR

Un administrador es un empleado de la empresa Pcware que tiene los mayores privilegios del sistema en desarrollo, y se encarga de gestionar usuarios, roles, seguridad y preferencias de usuarios.

AUXILIAR DE SOPORTE

Un auxiliar de soporte representa a una persona empleada de la empresa propietaria del sistema, el cual es responsable de recibir informes de problemas presentados para ciertos productos entrantes desde el centro de atención al cliente, también está encargado de entrevistar a clientes compradores en cuanto a un problema o más del producto adquirido por éste y ver los problemas más importantes para un determinado producto.

Este actor necesita el sistema para recibir informes de problemas entrantes desde el centro de atención al cliente, listar problemas mas sobresalientes para ciertos tipo de producto, entrevistar al cliente comprador sobre un determinado problema que presenta el producto comprado, actualizar el estado del problema, añadir comentarios y dirigir el problema al técnico.

AUXILIAR DE ATENCION AL CLIENTE

El auxiliar de servicio al cliente es un empleado de la empresa encargada de verificar los derechos del cliente, introducir nuevos problemas y averiguar el estado de los problemas en el proceso de soporte.

TECNICO

Un técnico es un empleado de la compañía encargado de recibir informes de problemas de soporte de cliente, ver los problemas mas importantes para cada producto, analizar y solucionar el problema, añadir comentario al informe del problema y dirigir el problema a control de calidad.

AUXILIAR CONTROL CALIDAD

Un controlador de la calidad es una persona encargada de recibir informes de un problema y sus soluciones, realizar test de funcionamiento, añadir comentarios al informe del problema, redirigir los problemas solucionados y revisados a soporte de cliente, para que sean despachados al cliente dueño del producto. También puede devolver el problema al tecnico si fallan los test.

3.6.1.2 Casos de Uso

Durante esta fase se han identificado y añadido los siguientes casos de uso:

- Validar usuario
- Inicio usuario
- Realizar cotización
- Solicitar producto
- Hacer pedido
- Comprar producto
- Buscar producto
- Ver descripción de producto
- Ingresar información de envio

DOCUMENTACION DEL CASO DE USO: VALIDAR USUARIO

Nombre del caso de uso: Validar usuario

Actor principal: Cliente, Administrador y director de ventas

Historia de la revisión:

Fecha	Versión	Descripción	Autor
14/03/2005	1.0	Versión Inicial	Jeovany Enrique López Polo

Descripción:

El caso de uso validar usuario(login) permite a los usuarios clientes, empleados, administrador y director de ventas entrar al sistema Tienda Pcware para utilizar alguno de sus servicios como usuarios registrados.

Precondiciones:
Este caso de uso no puede ser invocado directamente por un actor, sino que debe ser incluido por otros casos de uso.

Restricciones de despliegue:
Los clientes visitantes pueden navegar por los servicios publicos de la aplicación(Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

Flujo de eventos (Sucesos):

- El usuario ingresa su nombre de usuario y la clave. En caso de ser un usuario registrado, se crea una sesión de usuario.
- El usuario invoca al caso de uso para digitar su nombre de usuario y su clave.
- El usuario inserta su nombre de usuario y clave y hace submit a la información.
- La aplicación primero verifica que existe almacenado el nombre de usuario digitado.
- Si se encuentra el nombre de usuario, el sistema valida la clave digitada con la que se encuentra almacenada para este nombre de usuario.
- Si la clave es válida(coincide con la almacenada), el sistema determina en base al nombre de usuario si es un cliente persona, cliente compañía o ninguno, para establecer si ha comprado antes.
- Se carga la oferta vigente más conveniente para el tipo de cliente(persona compañia).
- La aplicación devuelve la siguiente información para el cliente almacenado: nombre completo, lista de productos comprados, lista de ofertas vigentes ordenadas por prioridad de conveniencia.
- La aplicación crea una sesión para el usuario y guarda en esta la información obtenida. Include(Inicio usuario)

Flujo alterno de evento:

Nombre de usuario no encontrado

- El siguiente mensaje es desplegado “Nombre de usuario o clave inválido, por favor intente otra vez”
- El sistema registra este intento fallido.
- Si el numero de intentos fallidos es mayor que cinco, el siguiente mensaje es desplegado “Cuenta desabilitada temporalmente por seguridad, por favor comunicarse con el administrador del sistema”.
- El caso de uso comienza de nuevo.

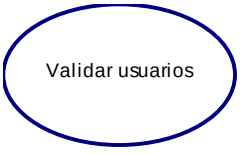
Clave inválida

- Si la clave no es válida (la digitada es diferente a la almacenada), el sistema despliega el siguiente mensaje “Nombre de usuario o clave inválido, por favor intente otra vez”.
- El sistema registra este intento fallido.
- Si el numero de intentos fallidos es mayor que cinco, el siguiente mensaje es desplegado “Cuenta desabilitada temporalmente por seguridad, por favor comunicarse con el administrador del sistema”.
- El caso de uso comienza de nuevo.

Cuenta deshabilitada

- La cuenta de usuario esta actualmente deshabilitada.
- El sistema registra este intento fallido.
- La aplicación despliega el siguiente mensaje “Cuenta actualmente desabilitada, por favor comunicarse con el administrador del sistema”.
- El caso de uso comienza de nuevo.

Figura 10. Diagrama del caso de uso: Validar Usuarios



DOCUMENTACION DEL CASO DE USO: INICIO USUARIO

Nombre del caso de uso: Inicio usuario

Actor principal: Cliente, Administrador y director de ventas

Historia de la revisión:

Fecha	Versión	Descripción	Autor
19/03/2005	1.0	Versión Inicial	Sandra Milena Maldonado

Descripción:
El caso de uso en base al tipo de usuario retorna la página de inicio para dicho rol, el cual es el papel con el que se está autenticando el usuario.

Precondiciones:
Este caso de uso no puede ser invocado directamente por un actor, sino debe ser incluido por otros casos de uso.

Restricciones de despliegue:
Los clientes visitantes pueden navegar por los servicios públicos de la aplicación(Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

Flujo de eventos (Sucesos):

- El caso de uso con el nombre de usuario y la clave retorna la página de inicio, de acuerdo al rol con el que se valida el usuario y los servicios a los que tenga derecho éste, en caso de no encontrarse ningún tipo de privilegio se retorna la página de inicio de un cliente visitante.
- Se invoca al caso de uso, pasándole un nombre de usuario y una clave.
- El sistema determina los roles.
- La aplicación determina la página de inicio para el rol con el que se está autenticando el usuario.

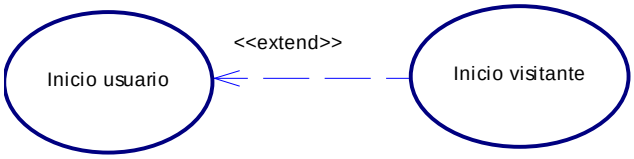
- El sistema retorna como punto de entrada dicha página de inicio, que puede ser inicio cliente visitante, inicio cliente registrado, inicio administrador o inicio de director de ventas.
- El caso de uso termina

Flujo alterno de evento:

Rol sin página de inicio

- La aplicación no encuentra la página de inicio para el rol con el que se está autenticando el usuario.
- El sistema retorna como punto de entrada la página de inicio del cliente visitante.

Figura 11. Diagrama del caso de uso: Inicio Usuario



DOCUMENTACION DEL CASO DE USO: SOLICITAR PRODUCTO

Nombre del caso de uso: Solicitar producto

Actor principal: Cliente

Historia de la revisión:

Fecha	Versión	Descripción	Autor
12/03/2005	1.0	Versión Inicial	Sandra Milena Maldonado

Descripción:

El caso de uso solicitar producto, permite a los usuarios clientes, entrar al sistema Tienda Pcware para hacer una solicitud de un producto que no encontró en las diferentes busquedas brindadas por el sistema.

Precondiciones:

El cliente no encontró la descripción del producto en las diferentes busquedas que se pueden hacer en el sistema.

El cliente ingresa como usuario registrado, es decir este servicio sólo esta disponible para un cliente que tenga una cuenta activa en el sistema.

Restricciones de despliegue:

Los clientes visitantes pueden navegar por los servicios públicos de la aplicación(Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

Flujo de eventos (Sucesos):

- El usuario puede haber ingresado como usuario registrado o estar de forma pública (cliente visitante), interactuando con los servicios de la aplicación, en el primer caso se crea una sesión de usuario.
- El usuario invoca al caso de uso para digitar la descripción del producto que desea solicitar.
- El usuario inserta la descripción del producto a solicitar, lo más detalladamente posible.
- El sistema solicita al cliente que ingrese con su cuenta de usuario activa.
- Con el nombre de usuario la aplicación determina la información personal del cliente, de la cual utiliza su email para el envio de la respuesta de la solicitud.
- El sistema presenta una patalla de confirmación donde se muestra la descripción del producto solicitado junto con los datos de envio de la respuesta a la solicitud, y aparecen las opciones de aceptar o cancelar.
- El cliente selecciona la opción de aceptar.
- La solicitud se registra en el sistema y queda en estado pendiente.
- El caso de uso termina.

Flujo alterno de evento:

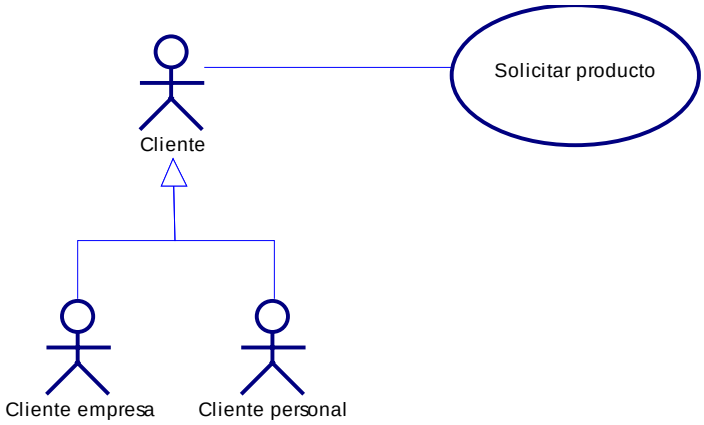
Cliente no registrado

- El cliente no tiene una cuenta en el sistema activa.
- El sistema le solicita al cliente que se registre, proceso en el cual captura la información de envio para las respuestas a las solicitudes hechas por éste.

Selección opción cancelar

- El cliente selecciona la opcion cancelar.
- El caso de uso comienza de nuevo.

Figura 12. Diagrama del caso de uso: Solicitar producto



DOCUMENTACION DEL CASO DE USO: REGISTRARSE

Nombre del caso de uso: Registrarse

Actor principal: Cliente

Historia de la revisión:

Fecha	Versión	Descripción	Autor
24/04/2005	1.0	Versión Inicial	Jeovany Enrique López Polo

Descripción:

Este caso de uso permite a un cliente cualquiera crear una cuenta de usuario válida en el sistema, con la cual, podrá acceder a los diferentes servicios que tiene la aplicación para los clientes registrados.

Precondiciones:

Ninguna.

Restricciones de despliegue:

Los clientes visitantes pueden navegar por los servicios públicos de la aplicación (Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

Flujo de eventos (Sucesos):

- El usuario ingresa su información personal al igual que un nombre de usuario y la clave, que será utilizada para autenticarlo, cuando quiera entrar a la aplicación como cliente registrado.
- El usuario invoca al caso de uso para registrarse en el sitio.
- El usuario ingresa su información personal, de la cual es requerida nombres, apellidos, email, información de envío, nombre de usuario y una clave y hace submit a la información.
- La aplicación primero verifica que no existe almacenado el nombre de usuario digitado por el cliente.
- Si no se encuentra el nombre de usuario, registra la información de usuario en el sistema.
- Se le presenta un mensaje de bienvenida que al cliente.
- Include (Inicio usuario).

Flujo alterno de evento:

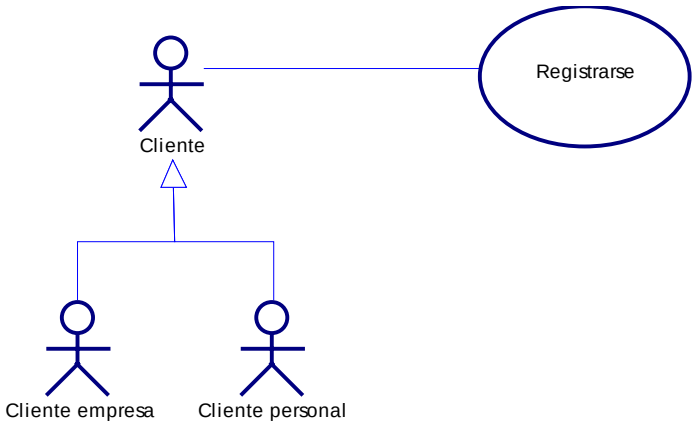
Dato requerido no digitado

- El cliente no digita uno de los datos requeridos
- El sistema lo retorna a la pantalla anterior desplegando un mensaje donde especifica que dato no ha sido digitado y que es requerido.

Nombre usuario ya existe

- La aplicación encuentra un usuario en el sistema con ese nombre
- El sistema regresa al cliente a la pantalla de captura de la información y le solicita que ingrese otro nombre de usuario y digite nuevamente la clave.

Figura 13. Diagrama del caso de uso: Registrarse



DOCUMENTACION DEL CASO DE USO: BUSCAR TIPO PRODUCTO

Nombre del caso de uso: Buscar tipo producto

Actor principal: Cliente

Historia de la revisión:

Fecha	Versión	Descripción	Autor
30/04/2005	1.0	Versión Inicial	Sandra Milena Maldonado

Descripción:
El caso de uso buscar tipo de producto permite a los usuarios clientes la busqueda de un tipo de producto (descripción de producto) por diferentes criterios.

Precondiciones:
Ninguna.

Restricciones de despliegue:
Los clientes visitantes pueden navegar por los servicios públicos de la aplicación (Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

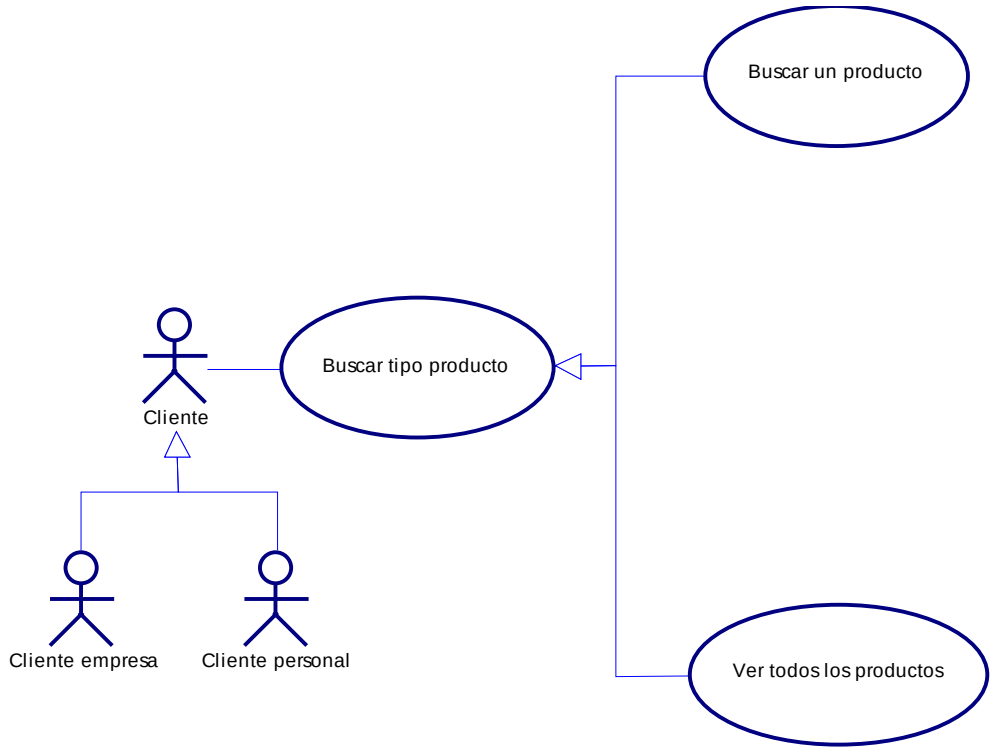
Flujo de eventos (Sucesos):

- El usuario ingresa un valor de búsqueda para un determinado tipo de producto, el sistema consulta que ocurrencias hay en los registros almacenados en la parte del nombre o la descripción y retorna todos los encontrados.
- El usuario invoca al caso de uso para digitar una cadena de búsqueda.
- El sistema busca que registros que tienen dicha cadena como parte del nombre o la descripción en la tabla de tipo producto.
- La aplicación retorna una colección con los registros encontrados.
- Se despliega dicha colección, donde para cada tipo producto encontrado se imprime el nombre, precio y descripción general.
- El caso de uso termina

Flujo alterno de evento:
No se encontró ningún tipo de producto:

- El sistema no encuentra ningún tipo de producto que coincida con la cadena digitada
- El siguiente mensaje es desplegado “No se encontró ninguna ocurrencia, intentar nuevamente”.
- Aparece la opción de solicitar el producto. include(Solicitar producto).

Figura 14. Diagrama del caso de uso: Buscar Un Producto



DOCUMENTACION DEL CASO DE USO: VER DESCRIPCION PRODUCTO

Nombre del caso de uso: Ver descripción producto

Actor principal: Cliente

Historia de la revisión:

Fecha	Versión	Descripción	Autor
14/05/2005	1.0	Versión Inicial	Sandra Milena Maldonado

Descripción**:**

El caso de uso ver descripción de producto permite a los usuarios clientes ver información ya sea general o detallada acerca de un producto en particular.

Precondiciones**:**

Se tiene por lo menos un tipo producto en la sesión de usuario registrado en el sistema.

Restricciones de despliegue**:**

Los clientes visitantes pueden navegar por los servicios públicos de la aplicación (Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

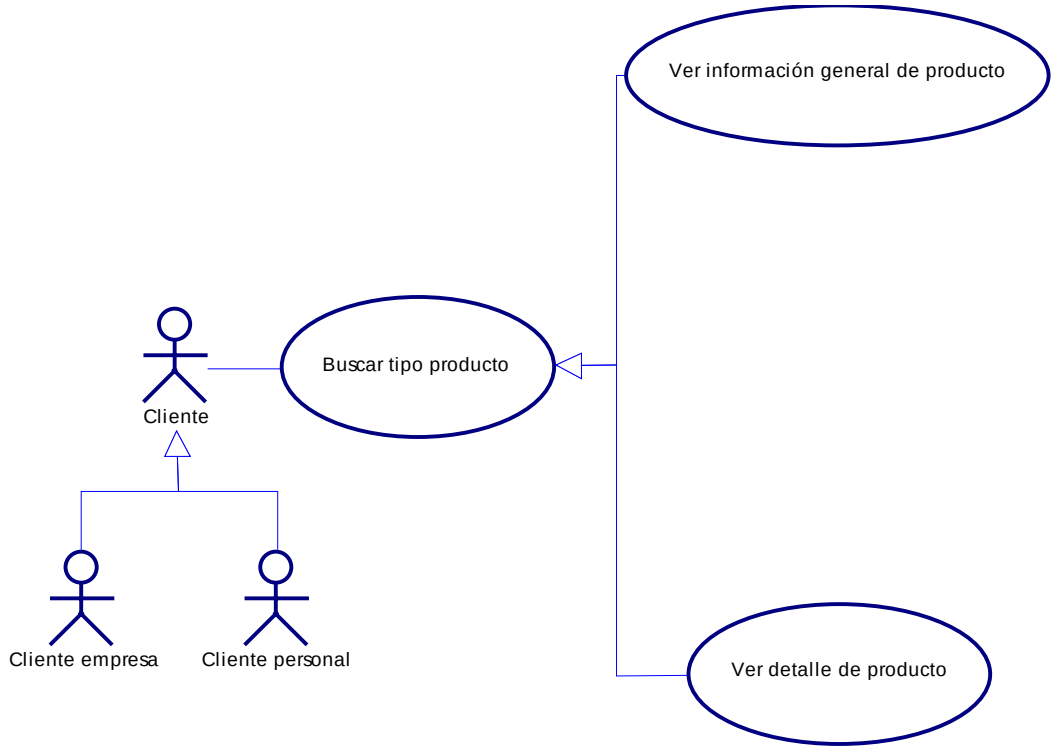
Flujo de eventos (Sucesos)**:**

- El cliente selecciona el tipo producto del cual quiere ver la decripción. El sistema despliega la información de forma general o detallada.
- El usuario invoca al caso de uso o es llamado por otro caso de uso.
- El cliente selecciona el tipo producto.
- La aplicación determina la información general del tipo de producto (nombre, precio y descripción general) con el identificador del tipo de producto.
- El sistema despliega la información que incluye nombre, precio y descripción general.
- Se presenta la opción de leer más acerca de este tipo de producto.
- El cliente selecciona la opción leer más.
- El sistema recupera el identificador del tipo de producto.
- La aplicación determina la información detallada para el tipo de producto en base al identificador.
- El sistema despliega la información que incluye nombre, precio y descripción detallada.
- El caso de uso termina.

Flujo alterno de evento**:**

No hay

Figura 15. Diagrama del caso de uso: Descripción producto



DOCUMENTACION DEL CASO DE USO: COMPRAR PRODUCTO

Nombre del caso de uso: Comprar producto

Actor principal**:** Cliente

Historia de la revisión:

Fecha	Versión	Descripción	Autor
27/06/2005	1.0	Versión Inicial	Jeovany Enrique López Polo

Descripción**:**

El caso de uso comprar producto permite a los usuarios clientes añadir al carro de compra uno de los producto ofrecidos por la tienda virtual Pcware.

Precondiciones**:**

Ninguna.

Restricciones de despliegue**:**

Los clientes visitantes pueden navegar por los servicios públicos de la aplicación (Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

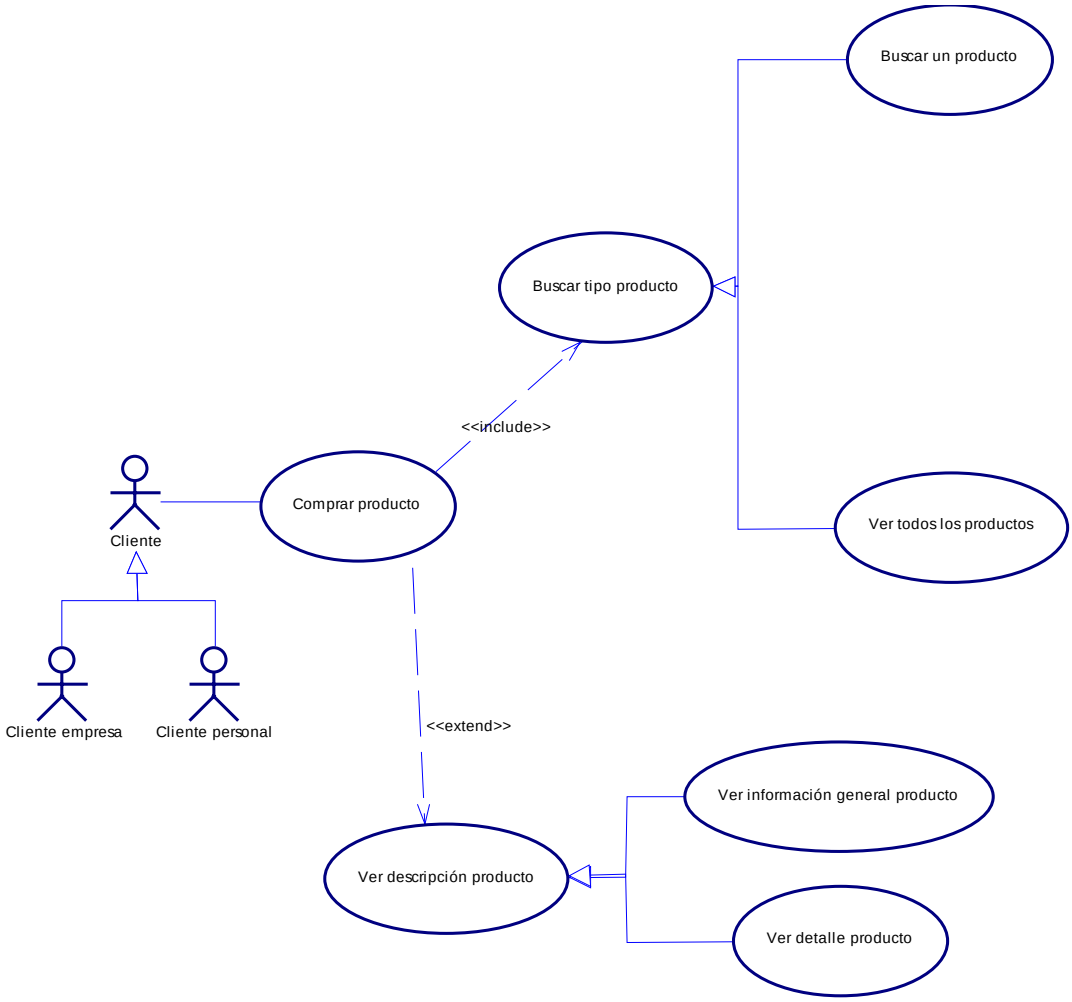
Flujo de eventos (Sucesos)**:**

- El usuario carga un producto a su carro de compra y la aplicación determina el nuevo total de lo que lleva en el carrito de compra.
- Incluye (Buscar tipo producto).
- Incluye (Ver descripción producto).
- El cliente selecciona la opción comprar producto.
- El sistema recupera de la peticion el identificador del tipo producto.
- Con el identificador del tipo de producto se busca la descripción general, que incluye el precio.
- Se carga al carro de compra que se encuentra en la sesión del tipo de producto recuperado.
- La aplicación calcula el nuevo subtotal de la compra.
- Se despliega al cliente un resumen de su orden de compra.
- Se presenta la opción de seguir comprando
- El cliente selecciona la opción seguir comprando
- El caso de uso comienza de nuevo

Flujo alterno de evento**:**

No hay

Figura 16. Diagrama del caso de uso: Comprar producto



DOCUMENTACION DEL CASO DE USO: HACER PEDIDO

Nombre del caso de uso: Hacer pedido

Actor principal: Cliente

Historia de la revisión:

Fecha	Versión	Descripción	Autor
05/07/2005	1.0	Versión Inicial	Jeovany Enrique López Polo

Descripción**:**
El caso de uso hacer pedido permite a los usuarios clientes solicitar una orden de pedido que incluye varios de los producto ofrecidos por la tienda virtual Pcware.

Precondiciones**:**
Ninguna.

Restricciones de despliegue**:**
Los clientes visitantes pueden navegar por los servicios públicos de la aplicación(Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

Flujo de eventos (Sucesos)**:**

- El usuario carga todos los tipos de productos que desea comprar, a su carro de compra y la aplicación determina el total de lo que lleva y registra esta orden de compra en el sistema. Incluye (Comprar producto).
- El cliente seleciona que la orden esta completa, despues de comprar uno o más productos.
- Aparece un resumen de la orden de pedido con descripción general de lo comprado con los subtotales y el total global.
- El cliente selecciona aceptar.
- El sistema detemina si hay un cliente en la sesión.

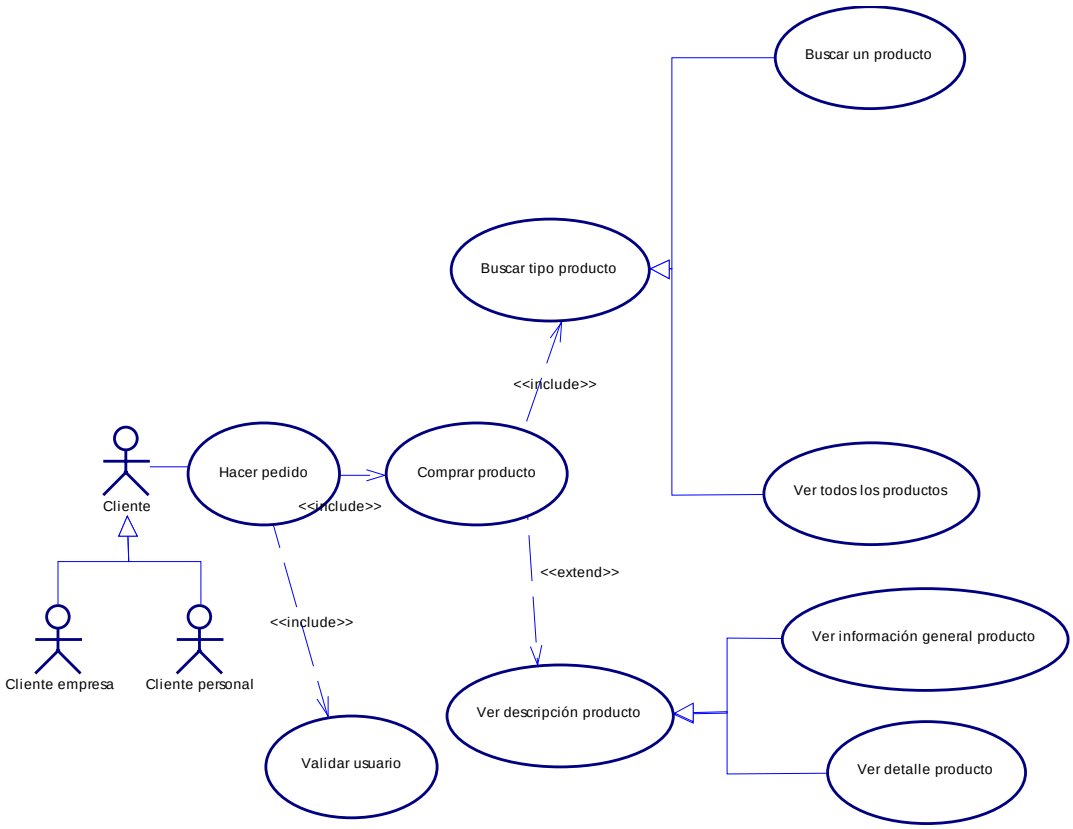
- Se recupera el cliente de la sesión.
- La aplicación solicita al cliente la información de envío de la orden.
- La orden es registrada para el usuario(cliente) en la sesión.
- Se despliega un mensaje que dice “Su orden ha sido registrada, espere pronta notificación para el despacho”
- El caso de uso termina

Flujo alterno de evento**:**

No hay usuario cliente en la sesión

- La aplicación determina que no hay un cliente en la sesión de usuario
- incluye(Validar usuario)

Figura 17. Diagrama del caso de uso: Hacer Pedido



DOCUMENTACION DEL CASO DE USO: GESTIONAR ORDENES DE PEDIDO

Nombre del caso de uso: Gestionar órdenes de pedido

Actor principal: Director ventas

Historia de la revisión:

Fecha	Versión	Descripción	Autor
25/07/2005	1.0	Versión Inicial	Jeovany Enrique López Polo

Descripción**:**
El caso de uso gestionar órdenes de pedido, permite al director de ventas de la empresa Pcware verificar el estado de una orden, autorizar su despacho así como recupera información del cliente para ponerse en contacto con él.

Precondiciones**:**
Ninguna.

Restricciones de despliegue**:**
El director de ventas pueden navegar por los servicios de la aplicación (Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

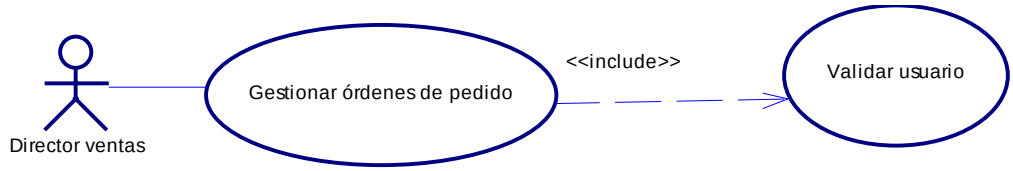
Flujo de eventos (Sucesos)**:**

- El sistema despliega todas las órdenes en estado vigente, el director selecciona una para ver los detalles.
- El sistema lista todas las ordenes registradas en estado vigente.
- El director selecciona una orden.
- La aplicación recupera el identificador de la orden.
- Se determinan los detalles de una orden en base al identificador.
- Se busca quien(cliente) emitió dicha orden.
- Se recupera el listado de todos los tipos de productos asociados a dicha orden.

- La aplicación despliega un resumen con toda la información recuperada.
- El director autoriza el despacho de la orden.
- El sistema registra dicho despacho y cambia el estado de la orden a despachada.
- El caso de uso termina

Flujo alterno de evento**:**
No hay

Figura 18. Diagrama del caso de uso: Gestionar órdenes de pedido



DOCUMENTACION DEL CASO DE USO: GESTIONAR USUARIOS

Nombre del caso de uso: Gestionar usuarios

Actor principal: Administrador

Historia de la revisión:

Fecha	Versión	Descripción	Autor
03/08/2005	1.0	Versión Inicial	Sandra Milena Maldonado

Descripción**:**
El caso de uso gestionar usuarios, permite al administrador de la empresa Pcware cambiar el estado (activa, deshabilitada, bloqueda) de una cuenta de usuario, crearla o eliminarla.

Precondiciones**:**
Ninguna.

Restricciones de despliegue**:**
El administrador pueden navegar por los servicios de la aplicación (Web Site) desde cualquier computador conectado a internet a través de un navegador cliente como Internet Explorer, Netscape, etc.

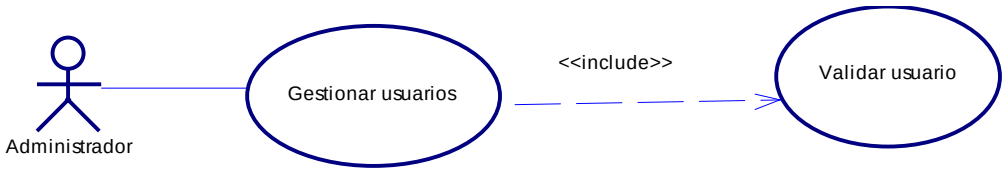
Flujo de eventos (Sucesos)**:**

- El sistema despliega todas las cuentas de usuario existentes, el administrador selecciona una para ver los detalles con la opción de poder modificarla o eliminarla.
- El sistema lista todas las cuentas de usuarios registradas en cualquier estado.
- El administrador selecciona una cuenta de usuario.
- La aplicación recupera el identificador de la cuenta.
- Se determinan los detalles de una cuenta de usuario en base al identificador.
- Se busca quien(usuario) es el propietario de dicha cuenta.
- La aplicación despliega el detalle con toda la información recuperada y la opción de modificar o eliminar.
- El administrador selecciona la opción de modificar.
- El sistema presenta la información de la cuenta con la lista de estados desprotegida.
- El administrador escoge el nuevo estado de la cuenta y la opción aceptar.
- La aplicación registra el cambio de estado para dicha cuenta de usuario.
- El caso de uso termina

Flujo alterno de evento**:**
El Administrador selecciona la orden eliminar

- El administrador selecciona la opción eliminar cuenta de usuario
- El sistema muestra una pantalla de confirmación con el detalle de la cuenta a ser eliminada
- El administrador selecciona aceptar
- La cuenta de usuario se elimina del sistema.

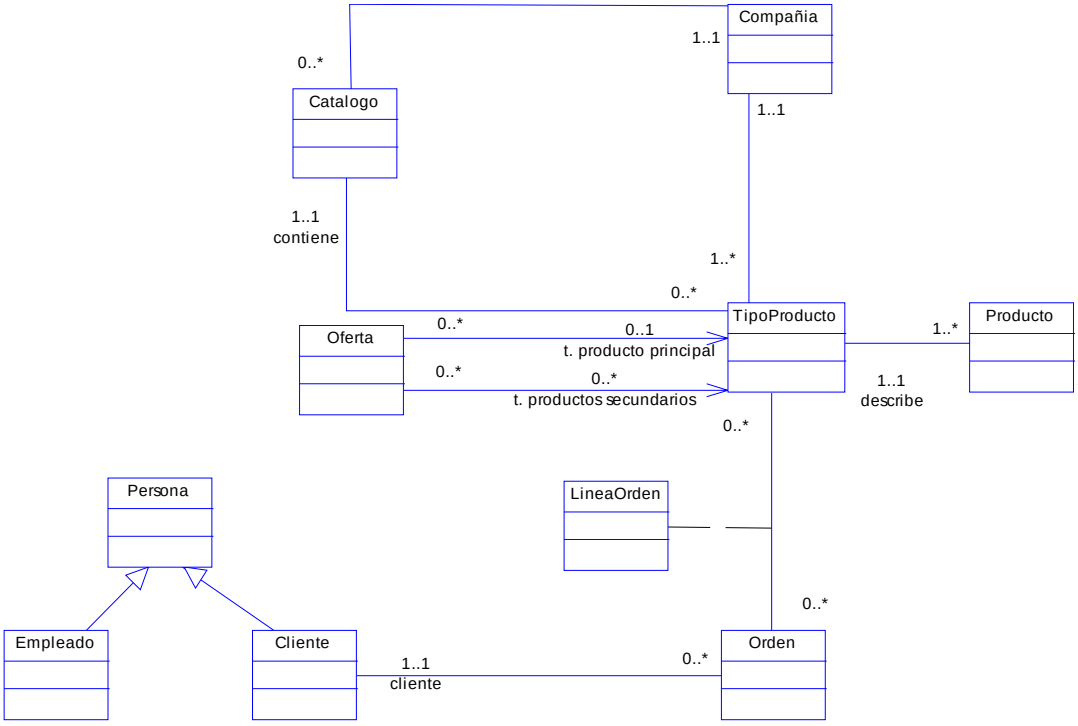
Figura 19. Diagrama del caso de uso: Gestionar usuarios



MODELO DE DOMINIO DE VENTAS DE PRODUCTOS REVISADO

De acuerdo al modelo de análisis realizado para la segunda iteración, el diagrama de clases queda complementado de la siguiente manera:

Figura 20. Modelo de dominio de ventas de productos



Las nuevas clases que aparecen en esta iteración se detallan a continuación:

Persona:

Representa a cualquier tipo de usuario de la aplicación, ya sea un cliente o empleado(como director de ventas, auxiliar soporte, etc) o inluso al mismo

administrador del sistema que para efectos de implementación se toma tambien como un empleado de la compañia.

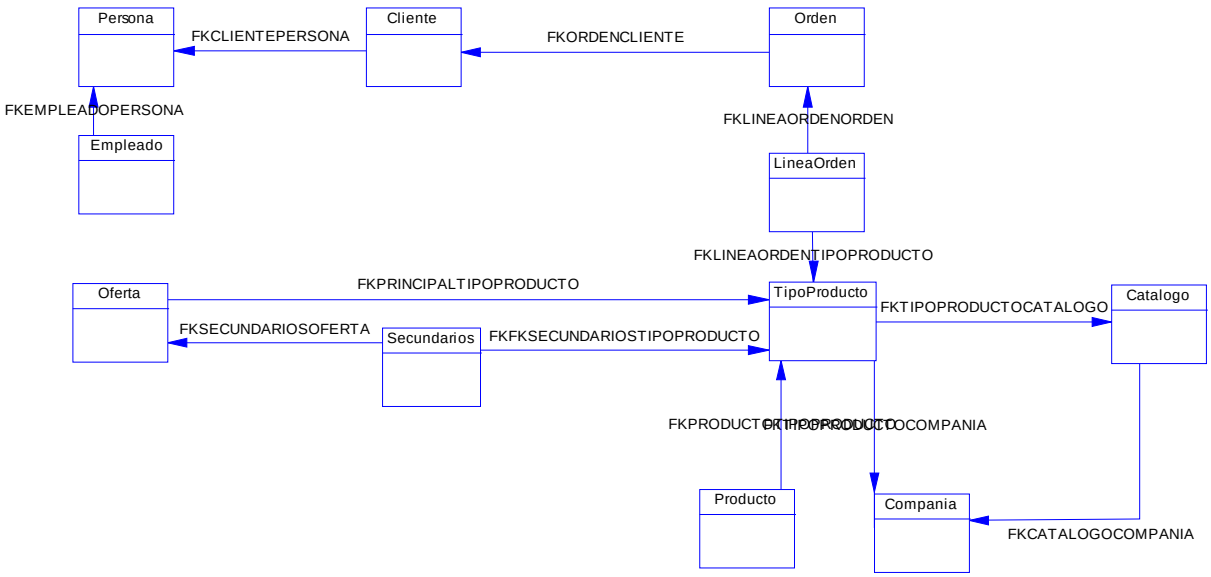
Empleado:
Representa persona que labora para la empresa propietaria del sistema, por simplicidad el usuario administrado se enmarca en esta clasificación.

Cliente:
Representa un persona que es un comprador potencial de la tienda, puede ser de dos tipo, cliente personal y cliente empresa.

Orden:
Encapsula todos los tipos de productos que realizó el cliente en un pedido dado; esta entidad se encarga de calcular totales para los diferentes pedidos.

LineaOrden:
Representa una entrada en una orden de pedido hecha por un cliente, de tal forma que tiene asociado un TipoProducto, un atributo llamado cantidad con lo cual puede calcular el total para esa linea de pedido.

Figura 21. Diagrama Entidad-Relación

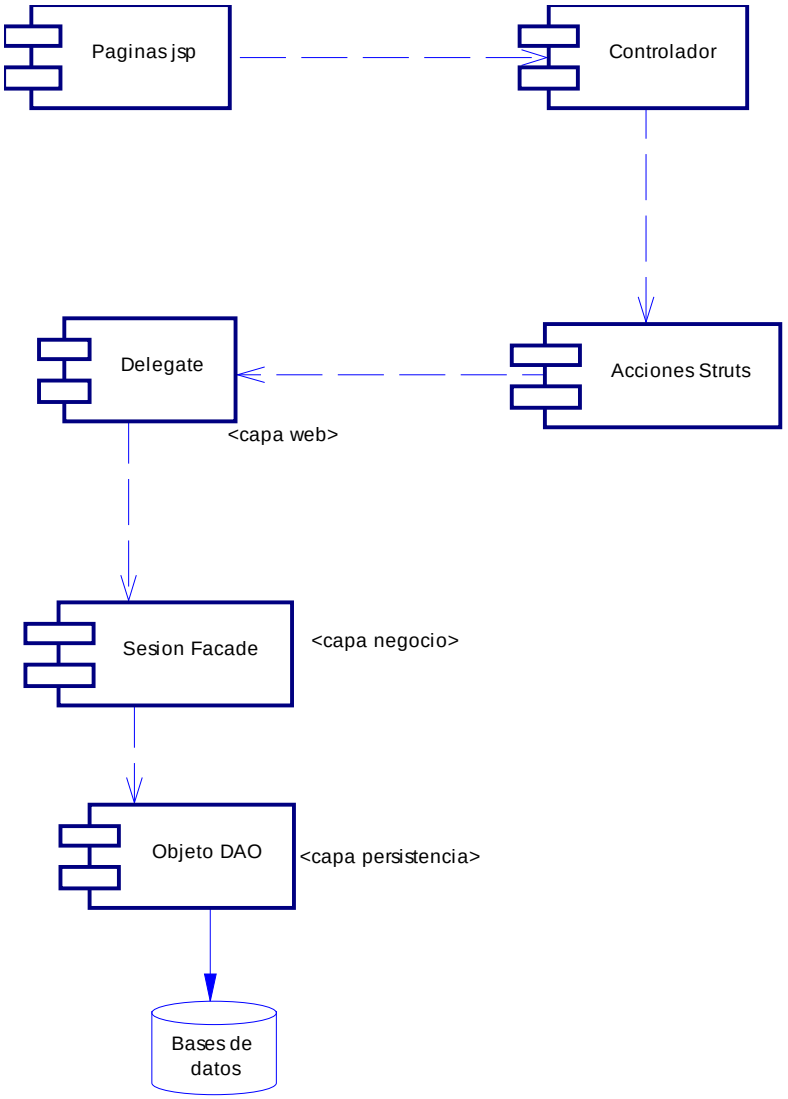


4. IMPLEMENTACION

4.1 DIAGRAMA DE COMPONENTES

A continuación se muestra el diagrama de componentes que describe la forma en la que se implementó los diferentes componentes que hacen parte de la aplicación

Figura 22. Diagrama de componentes



Paginas JSP:
Estas son utilizadas únicamente para desplegar información, no contienen ningún tipo de lógica.

Controlador:
Recibe todas las solicitudes de los usuarios y contiene la lógica necesaria para cargar dinámicamente las diferentes acciones asociadas a dichas solicitudes.

Acciones Struts:
Encapsula las acciones web que puede solicitar el usuario que se ejecutan para recibir un determinado servicio de la aplicación.

Delegate:
Es un componente que define una capa delgada entre las acciones web y la lógica del negocio(Sesion facade), su función es evitar que desde las acciones web se llame directamente a la lógica del negocio o sea al componente sesion facade.

Sesion facade:
Actua por simplicidad como un componente que contiene la lógica del negocio para la tienda virtual.

Objeto DAO:
Encapsula toda la lógica necesaria para recuperar o guardar información del negocio.

Bases de datos:
Utilizada para guardar la información de carácter persistente.

4.2 DESCRIPCION GENERAL DEL SITIO WEB PCWARE

El sitio Web PCWARE, es un medio de comunicación a través del cual, cualquier usuario con acceso a internet, pueda obtener información de catálogos de productos hardware, permitiéndole realizar cotizaciones y pedidos según sus propias necesidades

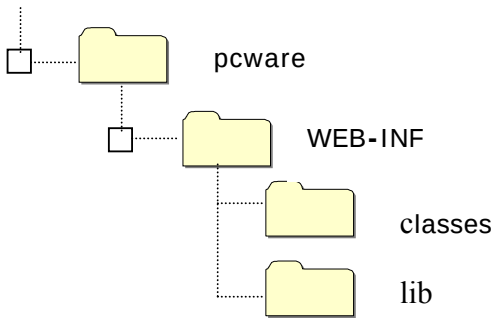
SERVIDOR DE PAGINAS WEB

El sitio Web reside en un servidor de páginas Web llamado Tomcat, el cual recibe las solicitudes hechas por los usuarios, las procesa, y les regresa como respuesta los documentos correspondientes usando para esto el protocolo http y el formato HTML, los cuales pueden ser visualizados por parte del usuario mediante navegación mediante un navegador de Internet.

Tomcat se diferencia de otros servidores Web, en que está capacitado para procesar las solicitudes que reciba usando las herramientas que le brinda el lenguaje de programación JAVA. Este servidor puede procesar solicitudes por contenido estático (documentos cuyo contenido no cambia de acuerdo a las solicitudes de algun usuario) y dinámico (documentos cuyo contenido es generado de acuerdo a solicitudes del usuario o consultando fuentes externas como bases de datos).

Tomcat es conocido como un contenedor de Servlets. Se encarga de recibir solicitudes y las transmite a aplicaciones Web hechas en Java. La aplicación Web recibe estas solicitudes, si son por un contenido estático, obtiene el documento y lo regresa a Tomcat, para que éste lo transmita al usuario, o si la solicituid es por contenido dinámico, es delegada al Servlet o página JSP (Java Server Pages) correspondiente que se encarga de procesarla y generar el documento HTML , el cual es transmitido al usuario.

Figura 23. Estructura básica de una aplicación Web



Todas la aplicaciones Web deben seguir esta estructura dada por la última especificación de JAVA (JSP 1.2 y Servlets 2.3). Sin embargo, el desarrollador es libre de agregar directorios mediante los cuales pueda organiar de forma lógica tanto los documentos estáticos, páginas JSP y otros recursos utilizados por la aplicación como imágenes, archivos de texto, etc.

El directorio WEB-INF es obligatorio y contiene la estructura de clases que utiliza la aplicaión, además de recursos de configuración. Dentro de esta carpeta se encuentra el archivo web.xml, el cual contiene diferentes opciones de configuración de la aplicaicón.

Este documento de tipo xml es leido por el servidor para ajustar algunas propiedades que puede tener la aplicaicón en particular. El archivo de la aplicación muestra el nombre de la aplición en la etiqueta llamada display-name.
Session-config es el valor en minutos por el cual un usuario puede mantener una sesión inactiva.
Welcome-file-list permite especificar nombres de páginas web que serán buscadas por el servidor cuando un usuario solicite únicamente el nombre de un directorio.

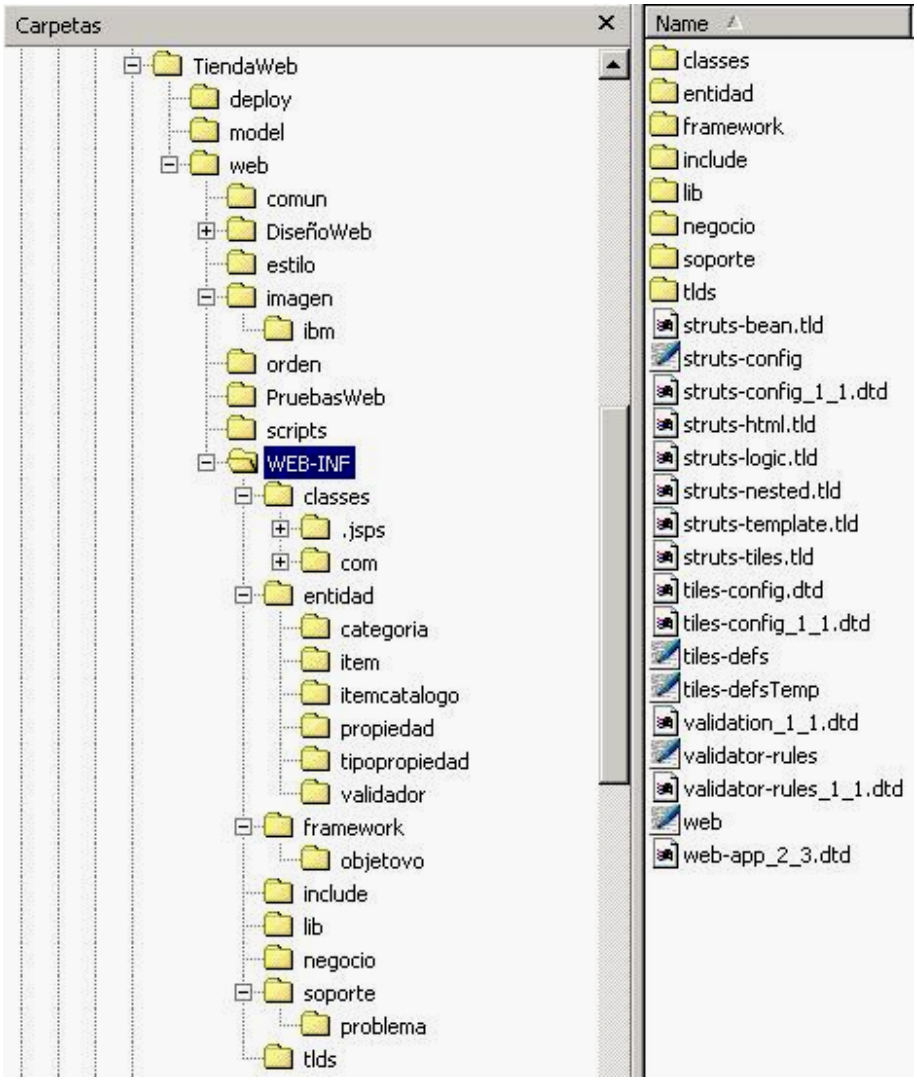
Dentro del directorio WEB-INF se encuentra el directorio “classes”, el cual contiene los archivos de clases compilados que use la aplicación en páginas JSP o los Servlets, y también se encuentra el directorio “lib” que contiene los archivos “.jar” que necesite la aplicación. Estos archivos son paquetes de clases compilados y comprimidos para su fácil transporte.

ESTRUCTURA DEL SITIO WEB

La estructura de directorios del sitio Web de PCWARE sigue las especificaciones del aplicaiones Web de JAVA, y por lo tanto usa la estructura mostrada en la sección anterior, agregánsole directorios adicionales que permitan una organización lógica del sitio.

En el directorio raíz llamada TiendaWeb, el cual se muetra en la siguiente figura, se encuentran una serie de páginas HTML y JSP, las cuales contienen información general acerca del proyecto PCWARE.

Figura 24. Estructura sitio Web PCWARE



La página principal o index.jsp, es el documento principal del sitio web, y es el punto de entrada para poder acceder a toda la información acerca del proyecto y los servicios que ésta presta.

Bajo la ruta WEB-INF/classes/com/, se encuentran las clases que usa el sistema de eventos para procesar las solicitudes realizadas por el usuario.

5. PRUEBAS

A través del desarrollo del proyecto, se hicieron prueba unitarias, con datos no reales, las cuales resultaron satisfactorias, permitiendo detectar posibles errores y corregirlos a tiempo.

No se realizaron pruebas de carga o ningún tipo de prueba piloto en la empresa, que involucraran datos reales.

6. CONCLUSIONES

La utilización de software gratuito como TomCat y Java permite a las empresas contar con herramientas que se caracterizan por brindar un excelente rendimiento y además con un bajo presupuesto se puede llegar a un buen término en el desarrollo de un proyecto.

Los servicios que ésta aplicación presta a los clientes, facilita el proceso de comercialización, siendo esto determinante en el aumento de las ventas

Mediante el desarrollo basado en casos de uso, se pudo capturar las funcionalidades principales de la aplicación.

La utilización de la arquitectura en tres capas permitió construir una aplicación robusta de forma que se puede extender fácilmente para agregar funcionalidades adicionales.

7. RECOMENDACIONES

De acuerdo a las inquietudes recogidas entre los empleados de la empresa Pcware y diferentes tipos de usuarios (clientes de dicha empresa) a lo largo del desarrollo del presente proyecto de grado, se recomienda:

Extender la aplicación de tal forma que pueda brindarse un soporte a usuarios y administración más completa de las diferentes entidades de una forma integral. Esto no se hizo posible porque estaba fuera del alcance del proyecto.

Integrar la presente aplicación con los sistemas existentes en la empresa Pcware en busca de evitar posibles inconsistencias sobre todo en la parte de cierres contables.

Implantar la aplicación en un servidor seguro, en lo posible sobre un sistema operativo Linux, con servidor web Apache que utilice a Tom-cat como contenedor de servlets, además definir políticas de seguridad a todos los niveles, desde el sistema operativo hasta las cuentas de usuarios en la base de datos.

Se recomienda establecer un plan de trabajo para la implantación del sistema. Además realizar reuniones periódicas con cada uno de los usuarios finales y directivas de la empresa involucradas en el desarrollo del proyecto, con el fin de ajustar cada vez más el sistema a las necesidades que se requieran para un buen desempeño.

Usar el presente sistema como línea base para futuros desarrollos, que busquen funcionalidades adicionales en dicha empresa.

BIBLIOGRAFIA

SHALLOWAY, ALAN y TROTT, James R. Diseño de Patrones, una nueva perspectiva en el diseño orientado a objetos. Addison Wesley. Madrid. 334p.

CEBALLOS SIERRA, Fco. Javier. Java tm 2, Curso de Programación. Alfaomega, México: 2000. 778p.

-----. El proceso unificado de desarrollo de software. Addison Wesley. Madrid: 2000. 464p.

RUNBAUGH, James. Modelado y diseño orientado a objetos. Prentice Hall. España: 1996. 643p.

BOOCH, Grady, RUMBAUGH, James y JACOBSON, Ivar. El lenguaje unificado de modelado. Addison Wesley. Madrid: 1999. 464p.

FALKNER, Jayson, et al. Beginning JSP web development. Wrox Press. 1 ed. 852p.

PRESSMAN, Roger. Ingeniería del software. Un enfoque práctico. Mc Graw-Hill. 5ed. 2001

REFERENCIAS EN INTERNET

http://jakarta.apache.org/tomcat/index.html Sitio oficial del servidor Web Tomcat.

http://java.sun.com Contiene una fuente completa acerca de las diferentes tecnologías Java.

http://webmaster.bankhacker.com/b2c/comercio-electronico.phtml Esta página Web, contiene conceptos generales sobre comercio electrónico y algunas guías o consejos para crear una tienda virtual.

http://www.tigerdirect.com Sitio o tienda virtual para la venta de partes y accesorios informáticos.

GLOSARIO

ATRIBUTO: Valor de un dato que es almacenado en los objetos de una clase. Cada atributo tiene un valor par una instancia.

AUTORIZACION: Proceso por el cual se determina a qué recursos tiene acceso cierto usuario identificado previamente.

CLASE: Una clase de objetos describe un grupo de objetos con propiedades (atributos) similares, con relaciones comunes entre otros y con una semántica común.

CLIENTE: En la arquitectura cliente- servidor, el cliente es el sistema o persona que a través de su computador solicita un servicio a un servidor, el cual está en la capacidad de dar respuesta a dichas solicitudes.

OBJETO: Concepto, abstracción o cosa con límites bien definidos y con significado a efectos del problema que se tenga entre manos. Tiene dos propósitos: promover la comprensión del mundo real y proporcionar una base práctica para su implementación por computadora.

VALIDACION: Es la actividad de comprobar que los atributos de un objeto permanezcan en el rango de valores admisibles por un sistema.

INSTANCIA: Es un objeto que es creado a partir de una clase determinada, elcual tiene atributos y operaciones propias.

METODO: Es una operación que define como se comporta un objeto.

JAVA: Lenguaje de programación que se caracteriza por tener una arquitectura que permite que el código escrito funcione en multitud de sistemas operativos sin ser modificado.

HTTP: (Protocolo de transferencia de Hipertexto) Protocolo que permite la transmisión de documentos de hipertexto entre el cliente que lo solicita y el servidor que lo suministra.

SERVIDOR: En la arquitectura cliente servidor, es el sistema que recibe solicitudes por parte de los clientes, las procesa y les regresa repuestas como resultado a las solicitudes.

SERVIDOR WEB: Sistema que tiene como objetivo recibir las solicitudes que un cliente hace a través de internet, para procesarlas y generar documentos o páginas Web como resultado.

SQL: (Structure Query Language) Lenguaje de cuarta generación que permite ejecutar operaciones como consultas, y acualizaciones sobre las entidades existentes en bases de datos relacionales.

E-comerce: (Comercio Electrónico) Nueva forma de comercializar productos, a través de un sitio Web.