UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
"ACATLAN"

"DISEÑO DE UNA APLICACION DE COMERCIO ELECTRÓNICO CON LA EXTENSION WAE (WEB APPLICATION EXTENSION) DE UML."

CASO PRÁCTICO:
TIENDA VIRTUAL FASHION POL.

SEMINARIO TALLER EXTRACURRICULAR

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

PRESENTA:
JUANA URSULA HERNANDEZ MORGADO

ASESOR:
LIC. MARITZA NOVA JUAREZ.

MARZO, 2004.

A:

Mis padres:
Humberto y Eustaquia, por todo su amor y esfuerzo.

Mis hermanos:
Alejandro, Elías, Inés y Oscar, por su cariño y ejemplo.

Los profesores:
Sara Camacho, por su entusiasmo y sus consejos.
Maritza Nova y Antonio Gama, por su confianza.
Rubén Romero y Carlos Rangel por su apoyo.

Y a todos los que han contribuido a mi formación como profesionista y ser humano.

¡ G R A C I A S !

A Javier:

Por el tiempo juntos, compartiendo sueños y realidades...
Por tu apoyo.
Por tu amistad y franqueza.
Y por todo, absolutamente todo.

A Lilián Yolotlí:

Por toda la alegría que has dado a mi vida.
Por tu ternura y confianza en mí.

Porque éste sueño, no sería realidad sin su apoyo.

¡ G R A C I A S !

DISEÑO DE UNA APLICACIÓN DE COMERCIO ELECTRÓNICO CON LA EXTENSION WAE (WEB APPLICATION EXTENSION) DE UML.
CASO PRÁCTICO:
TIENDA VIRTUAL FASHION POL.

ÍNDICE

INTRODUCCIÓN.

Este trabajo tiene como objetivo el presentar la extensión WAE (WEB Application Extension) de UML (Lenguaje Unificado de Modelado) como una alternativa para el diseño de aplicaciones WEB.

¿Puede una mediana empresa, que se dedica a la fabricación y venta de ropa de mezclilla, proyectar su marca en Internet y en consecuencia incrementar sus ventas? El que las prendas de vestir no se consideren entre los artículos que más se venden en Internet, no significa que una empresa de este tipo no deba molestarse en explorar las posibilidades que la red ofrece, una tienda en Internet no pretende venderle a todo el mundo, su principal objetivo es mostrarse ante una audiencia bien definida, a la cual, están dirigidos sus productos.

Una vez que una empresa ha tomada la decisión de introducir Internet en los procesos de su negocio, ¿qué debe considerar para tener éxito? Los factores a considerar son muy diversos, desde el diseño de la tienda, hasta las estrategias de publicidad para la tienda. El presente trabajo se centra en los factores de diseño y se enfoca al uso de la extensión WAE de UML; utilizando Rational Rose 2000 Enterprise Edition para la elaboración de los planos del software a construir de la tienda virtual.

La estructura de este trabajo puede dividirse en tres partes, la primera es un marco teórico que nos introduce a los conceptos de Internet, las aplicaciones y su ciclo de vida, destacando la importancia de la etapa del diseño; la segunda parte nos da una introducción al Lenguaje Unificado de Modelado y a su extensión de notación WEB Application Extension; la última parte muestra la situación actual de la empresa denominada Fashion Pol, a partir de esta información se definen los requisitos para la aplicación a desarrollar, concluyendo con un modelo "escrito" usando UML y la extensión WAE. La distribución del trabajo queda determinada de la siguiente forma:

El Capitulo 1: Aplicaciones WEB. Tiene como objetivo proporcionar al lector un panorama general acerca de las aplicaciones WEB. En este capitulo se describen algunos conceptos básicos acerca de Internet y las aplicaciones WEB, los diferentes tipos de aplicaciones que existen, la forma en que han evolucionado Internet y las aplicaciones, además de describir los protocolos de comunicación, factores de riesgo y estrategias de seguridad más ampliamente utilizadas, dando mayor importancia a las aplicaciones de comercio electrónico.

El Capitulo 2: Diseño de Aplicaciones WEB. Tiene como objetivo identificar la importancia de la fase del diseño dentro del ciclo de vida del software para Aplicaciones WEB. En este capitulo se analizarán algunos modelos del ciclo de vida del software destacando la importancia de la etapa del diseño para el éxito de un proyecto. Adicionalmente introduce los conceptos de arquitectura de aplicaciones y modelo de experiencia del usuario (UX); utilizados en el diseño de aplicaciones WEB.

El Capitulo 3: Extensión WAE para UML. Tiene como objetivo identificar la trascendencia de la extensión WAE de UML, dentro de la fase de diseño de aplicaciones WEB. En este capitulo se dará una visión general del Lenguaje Unificado de Modelado, incluyendo su historia, mostrando las ventajas que ofrece para visualizar, especificar, construir y modelar los artefactos de un sistema con gran cantidad de software, describiendo sus principales elementos y reglas, así como las especificaciones de diseño con la extensión WAE.

El Capitulo 4: Caso Práctico: Tienda Virtual: "FASHION POL". Tiene como objetivo aplicar la extensión WAE de UML para el diseño de una tienda virtual de la empresa FASHION POL. En este capitulo, se mostrará el uso de UML y su extensión WAE para la fase de diseño de aplicaciones, destacando sus ventajas de comunicación con el cliente y con los desarrolladores; los primeros podrán visualizar sin necesidad de conocimientos especializados si el diseño que se le presenten satisfacen o no sus requerimientos, y los segundos tendrán la información técnica suficiente para la construcción de la aplicación.

Capitulo 1:
APLICACIONES WEB.

1.1 Antecedentes

La reciente revolución en las tecnologías de la información ha modificado muchos hábitos de nuestra vida cotidiana; como por ejemplo, la forma de comunicarnos, la manera de hacer negocios, la forma de consultar información, entre otros.

Independientemente de nuestra opinión acerca de Internet, podemos asegurar que es una tecnología que ha tenido un gran impacto en nuestras vidas, tanto en lo político, lo social, lo cultural, e inclusive en lo privado de cada día. Esta forma de comunicación ha llamado la atención de investigadores, inversionistas, y hoy sabemos que también de millones de personas.

Pero, ¿cómo y cuándo surge Internet?, ¿cómo ha evolucionado para ser como hoy la conocemos? y ¿cómo ha variado en el tiempo la forma de comunicarnos con las computadoràs?.

1.1.1 Historia de Internet.

El comienzo del uso de las comunicaciones globales, se remonta a 1957, con la puesta en órbita del primer satélite artificial de la antigua Unión Soviética, el SPUTNIK.

En respuesta a este hecho en Estados Unidos se crea el Organismo de Proyectos de Investigación Avanzada (ARPA: Advanced Research Projects Agency), con el fin de establecer su liderazgo en el área de la ciencia y la tecnología aplicada a las fuerzas armadas.

Entre los estudios técnicos que se pueden destacar por sus aportaciones podemos mencionar en 1961, el primer trabajo de "conmutación de paquetes" de Leonard Kleinrock; en 1962 J.C.R. Licklider expone su concepto de "Red Galáctica", en esencia un concepto muy parecido a la Internet actual; en 1966 por Laurence G. Roberts "hacia una Red Cooperativa de Computadoras de Tiempo Compartido, de la cual presentó su primer diseño en 1967.

Para octubre de 1969 se concluyó la instalación de las primeras cuatro computadoras interconectadas de ARPANET, y se transmitió el primer paquete de datos. Estas primeras computadoras estaban situadas en la Universidad de California en Los Angeles (UCLA), Stanford Research Institute (SRI), Universidad de California en Santa Barbara (UCSB) y la Universidad de Utah (UU).

Para la siguiente década el crecimiento de ARPANET fue muy lento, pues no existían las herramientas necesarias para una comunicación rápida fiable y económica.

En 1971, Ray Tomlinson inventa el primer programa de correo electrónico y en 1972 lo modifica para ARPANET, eligiendo el símbolo @ de entre los símbolos de puntuación de una máquina de teletipos, para representar la palabra "*en*".

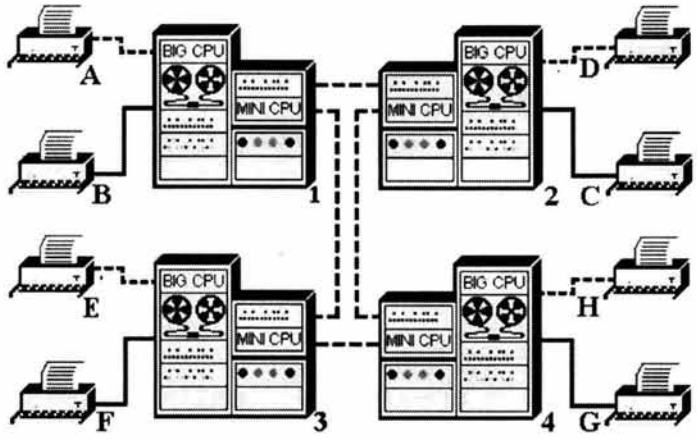

Figura 1-1: Primeros cuatro ordenadores interconectados.

También en 1971 la compañía Intel presenta el chip 4004, toda una unidad de calculo construida en un solo chip.

En 1973, se logra la primera conexión internacional de ARPANET.

En 1974 Vinton Cerf y Bob Kahn publican un protocolo para una red de intercomunicación de paquetes especificando en detalle el diseño del Programa de Control de Transmisión (TCP).

En 1975, Edward Robert presenta el primer microprocesador basado en un nuevo chip de Intel, el 8080.

Otro acontecimiento decisivo es el surgimiento de la primera computadora personal, la PC de IBM en 1981, ese año, sólo en Estados Unidos se vendieron 672,000 PC´s.

En 1982 se establecen el Protocolo de Control de Transmisión (TCP) y el Protocolo de Internet (IP) como el conjunto de

protocolos, conocido comúnmente como TCP/IP, para ARPANET este acontecimiento da el nombre actual a la red, hasta 1982 internet (con "i" minúscula) es utilizado para referirse a redes interconectadas, en general Internet (con "I" mayúscula) se utiliza desde 1982 para referirse a redes interconectadas usando TCP/IP.

En 1984 se introduce el DNS (Domain Name System), un sistema que normaliza el empleo de los nombres y direcciones en la red.

En 1989, Tim Berners-Lee escribe una propuesta de gestión de la información para distribuirla por medio de un sistema de hipertexto, al año siguiente reformula su propuesta y con ayuda de Robert Cailliau escribe una nueva propuesta que recibe el nombre de World Wide WEB (WWW), ese mismo año se desarrolla el primer navegador que permite enlazar las páginas. El WWW define una serie de normas para poder adjuntar imágenes, sonidos y otros elementos dentro de documentos escritos en un formato estandarizado para ser leído por cualquier usuario en la red. Este formato estándar se asienta en el protocolo HTTP (Hyper Text Transfer Protocol) y el lenguaje HTML (Hyper Text Markup Language).

En febrero de 1989 se conecta la primera máquina a Internet bajo el dominio .mx: dnx.mty.itesm.mx.

En 1990 deja de existir ARPANET, además se pone en línea el primer proveedor comercial de acceso telefónico a Internet.

En 1994, aparecen los shopping mall en Internet, claramente Internet se convierte en un medio comercial, ya que este año se

refleja un crecimiento explosivo de la WWW, y por primera vez se registran más dominios ".com" que dominios ".edu". .

Para 1995, Sun lanza Java Real Audio, una tecnología de audio que permite recibir sonido casi en tiempo real. La WWW avanza y se convierte en el servicio más importante de Internet, también es un año para el surgimiento de los motores de búsqueda.

El año clave para el comercio electrónico es 1998, cuando se instalan miles de tiendas virtuales donde se ofrecen todo tipo de productos.

Internet ha crecido y sigue creciendo más allá de las expectativas de su creación, son muchos los avances tecnológicos que han contribuido a fortalecerla, sin embargo, su historia es demasiado joven, y aun se esta escribiendo, de forma que no se puede decir que Internet ha terminado con su proceso de cambio. De hecho Internet debe continuar cambiando y evolucionando para mantenerse como un elemento relevante.

1.1.2 Evolución de las Aplicaciones.

Un programa o software de computadora es un conjunto de instrucciones que indica a la computadora, por medio de un lenguaje que la máquina interpreta o "comprende", como realizar una tarea paso a paso. Sin embargo, para poder comprender lo que es el software, es importante resaltar algunas de sus características:

- El software no es un elemento físico del sistema, es lógico.
- El software no se fabrica, se desarrolla.
- El software no se estropea, se deteriora.

Existen dos clases básicas de programas: del sistema y aplicaciones. Los programas del sistema ayudan a la computadora a realizar sus tareas operativas elementales y a su vez se dividen en sistemas operativos, controladores de dispositivos, utilerías y lenguajes de programación. Los programas de aplicación están diseñados y escritos para realizar tareas específicas.

Durante los primeros años (1950 a la primera mitad de la década siguiente) el software se diseñaba a la medida y tenía una distribución bastante pequeña, normalmente era la misma persona u organización que lo escribía, quien lo utilizaba, y si fallaba, lo corregía.

En la segunda era de la evolución del software (primera mitad de la década de los sesenta hasta finales de los setenta), los sistemas multiusuario modificaron los conceptos de interacción hombre-máquina, generando un nuevo mundo de aplicaciones; los avances en los dispositivos de almacenamiento condujeron a la primera generación de sistemas de gestión de bases de datos.

Durante la tercera era (mediados de la década de los setenta hasta una década después) se incremento la complejidad de los sistemas informáticos con los sistemas distribuidos, redes, comunicaciones digitales, así como la llegada y amplio uso de los microprocesadores.

La cuarta generación se caracteriza con la aparición de computadoras cada vez más pequeñas, pero más potentes, que se controlan con sistemas operativos sofisticados, interconectados tanto en redes globales como locales, las cuales soportan

aplicaciones de software avanzadas, empezando a cambiar la arquitectura de entornos centralizados de grandes computadoras a entornos descentralizados cliente/servidor.

La evolución de las aplicaciones va de la mano con la evolución de los lenguajes de programación, al principio la comunicación con las computadoras era por medio de tarjetas perforadas y programar una computadora era una tarea poco menos que imposible. Hacer un programa traía consigo la utilización de una cantidad considerable de tiempo, pero, esto se ha modificado a partir de la aparición de los entornos visuales de desarrollo.

Los primero años - Orientación por lotes - Distribución limitada - Software a la medida	Segunda Era - Multiusuario - Tiempo real - Bases de datos - Software como producto
Tercera Era - Sistemas distribuidos - Incorporación de inteligencia - Hardware de bajo costo - Impacto en el consumo.	Cuarta Era - Sistemas personales potentes - Técnicas orientadas a objetos - Sistemas expertos - Redes neuronales artificiales - Computación en paralelo - Redes de computadoras

Tabla 1-1: Evolución del Software

1.2 Conceptos Básicos.

Un par de computadoras se consideran una red, si están conectadas de tal forma que puedan compartir recursos, **Internet**, esta formada de diversas redes internacionales unidas

entre sí, por lo tanto podemos decir que es una red global (o una red de redes) que permite compartir recursos.

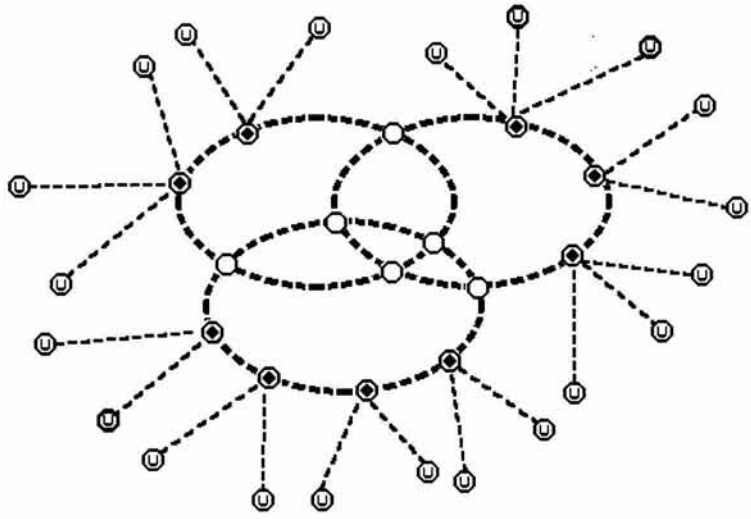

Figura 1-2: Estructura Simplificada de Internet.

La gráfica muestra de manera simplificada la estructura de Internet, donde cada elipse es una red más o menos extensa.

Los círculos vacíos representan ordenadores especiales denominados encaminadores (routers) que son puntos de interconexión.

Los círculos con punto representan proveedores de acceso a Internet.

Los círculos con una U, son los usuarios de Internet.

1.2.1 World Wide WEB (WWW).

El **World Wide WEB (WWW)**, o simplemente **WEB** no es el único servicio de Internet, pero si es el más utilizado, quizá, esto

se debe a su simplicidad y vistosidad, además contiene prácticamente todas las posibilidades de otros servicios.

El WWW esta formado por **Servidores WEB,** que son computadoras conectadas entre sí usando Internet, cada servidor cuenta con un **dominio** o nombre propio que es una dirección única e irrepetible, de acuerdo al Sistema de Nombre por Dominios **DNS (Domain Names System),** y que corresponde a una **dirección IP**[1]. Además, cada usuario posee un nombre propio que lo distingue, denominado **nombre de usuario.**

Las **páginas de Internet** son los documentos mostrados en nuestras pantallas al visitar lugares en Internet; y al conjunto de códigos usados para crear páginas de Internet, se le conoce como **HTML** (HiperText Mark-Up Language), en cada página podemos encontrar texto o gráficos que tienen vínculos incrustados, es decir **hipertexto,** que nos permiten desplazarnos fácilmente por la WEB.

Cada lugar o página en Internet posee una **dirección URL (Uniform Resource Locator),** que es una forma compacta de dirigirnos a una página WEB desde cualquier parte del mundo.

Se dice que un **WEB Site** (Sitio WEB), es un conjunto de páginas WEB contenidas en un Servidor WEB, a las cuales un

---

[1]Las direcciones IP están compuestas por cuatro grupos de ocho bits con valores de 0 a 255 cada uno, es decir cuatro grupos de tres dígitos cada uno; por ejemplo: 147.96.21.31, y permiten identificar de manera única dispositivos conectados.

usuario de Internet puede acceder, por medio de una página principal.

1.2.2 Comunicación en Internet.

Los equipos informáticos conectados a Internet utilizan un conjunto de reglas estándar establecidas mediante un **protocolo**, que describe el formato que los mensajes deben tomar, y la manera en que las computadora deben intercambiar mensajes dentro un contexto determinado, así algunos de los protocolos usados en Internet son **TCP/IP**, **FTP** (File Transfer Protocol) y **HTTP**(Hyper Text Transfer Protocol).

1.2.3 Servicios WEB.

Se denominan servicios a las posibilidades que ofrece Internet, cada servicio es una forma de sacarle provecho a la Red, y es independiente de los demás. Los servicios de Internet están basados en la relación Cliente/Servidor.

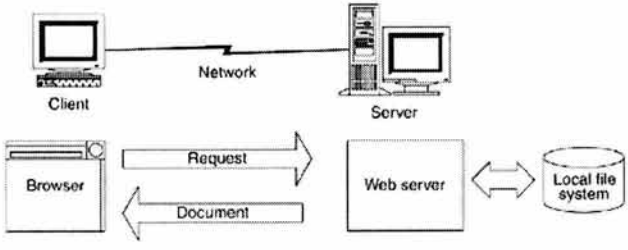

Figura 1-3: Estructura Cliente/Servidor

Un servidor es una computadora que ejecuta acciones para otra computadora, y el cliente es la computadora que solicita la acción.

En esta relación que ofrece muchas posibilidades, es de destacarse que al cliente no le interesa la forma en que el servidor realiza la tarea solicitada, únicamente le interesa el resultado de la tarea.

Internet ofrece muchas posibilidades de servicios, algunos de ellos contenidos, dentro de la WEB, tales como:

- Correo Electrónico, que permite enviar y recibir mensajes en formato electrónico de diferentes partes del mundo, sus ventajas son la velocidad de comunicación, así como el tiempo y el costo;

- Conexión Remota (TELNET), que permite la conexión a otra máquina, tiene la limitación de trabajar en modo texto, con comandos específicos, pero, es muy útil para consulta de bases de datos.

- FTP, esta herramienta hace posible acceder a documentos de una máquina remota, y traerlos a nuestra máquina.

1.3 Tipos de Aplicaciones WEB.

1.3.1 Aplicaciones WEB.

Es difícil dar una definición precisa de Aplicaciones WEB, podríamos decir simplemente que se trata de software en la WEB, También se les conoce como WEB APP ó Aplicaciones basadas en WEB.

Las aplicaciones WEB usan tecnologías capaces de hacer sus contenidos dinámicos, y permitir a los usuarios modificarlos en

el servidor. La diferencia entre una Aplicación WEB y un WEB Site, es muy sutil y esta dada por la capacidad que tiene un usuario de modificar el estado del servidor.

Una aplicación WEB es un tipo específico de WEB Site que de forma implícita y explícita almacena y manipula datos únicos para cada uno de sus usuarios; se diferencian de los tradicionales sitios basados en contenido, en que una aplicación WEB interactúa con los usuarios de una manera personalizada, cada usuario requiere identificarse y proporcionar una contraseña, una vez identificado, cada usuario puede realizar cambios a sus datos creando, corrigiendo, y suprimiendo información almacenada en el servidor por medio de browsers.

Una aplicación WEB es un sistema WEB, que es un concepto relativamente nuevo en el mundo de la computación y que se desarrolla con lógica de negocios. Un sistema WEB esta formado por documentos de hipertexto. Los principales elementos de un sistema WEB son el cliente browser, la red y el servidor WEB.

En general podemos decir que una Aplicación WEB es un WEB Site donde el usuario introduce información durante la navegación a través del sitio y los datos introducidos afectan el estado del sitio; más allá, por supuesto, de los registros del acceso y contadores de golpe. Esencialmente, una Aplicación WEB utiliza un WEB site como front-end de una aplicación de negocio.

1.3.2 Herramientas de las Aplicaciones WEB.

Una aplicación WEB esta basada en un modelo Cliente/Servidor. Entre clientes y servidores, el principal protocolo de

comunicación es el HTTP, que es un protocolo sin conexión, donde cada vez que el cliente requiere un documento de un servidor WEB, se debe establecer una nueva conexión. Para referirse a los recursos en la red como páginas WEB, imágenes, applets y otros recursos en la WEB, se utilizan los identificadores de recursos URL o URI (Uniform Resource Identifier)[2]. Para acceder a la información de Internet lo hacemos a través de programas que son capaces de leer los documentos, se denominan navegadores (browsers).

El principal lenguaje para dar formato a las páginas WEB es el HTML, que le dice al navegador como distribuir el contenido de la página en la pantalla.

Algunos de los archivos que pueden insertarse en un documento HTML son:

- Imágenes en formato *.gif, *.jpg, *.bmp, *.jpcg.
- Objetos de JavaScript.
- Plug-Ins.
- Archivos Flash.
- Archivos Shockwave.
- Videos en formato *.avi.

Las forms (formas) permiten a los usuarios introducir datos a través de páginas WEB.

Los frames (marcos) dividen las páginas WEB en regiones múltiples, permiten presentar información en una manera

---

[2] Identificador Unificado de Recursos, es una cadena que indica un recurso sobre el que aplicara un método solicitado.

flexible y útil, haciendo posible a un usuario tener activas al mismo tiempo múltiples páginas WEB.

Las cookies son pequeñas cantidades de datos que son enviados desde el servidor al cliente y se almacenan en el disco duro de su computadora, permite identificar las visitas a un sitio, o bien, pueden contener datos personalizados que le permiten al servidor "recordar" las preferencias de un usuario particular.

Algunas de las tecnologías con ambientes del desarrollo que proporcionan la infraestructura para la construcción de aplicaciones WEB son: CGI, Active Server Pages, JavaServer Pages, Servlets, y Server API's.

1.3.3 Tipos de Aplicaciones WEB.

Dentro de la diversidad de aplicaciones WEB disponibles para los consumidores destacamos las siguientes:

- Aplicaciones de Comercio Electrónico: Son las Aplicaciones WEB más populares, que permiten al usuario adquirir bienes y/o servicios.
- Servicios turísticos.
- Servicios Financieros.
- Portales de Información: Con contenidos comerciales o educativos.
- Servicios en Línea.

1.4 Comercio Electrónico.

1.4.1 Evolución.

Con el objetivo de hacer más flexible el comercio, la era electrónica introdujo nuevas formas de pago, diferentes a los tradicionales billetes y letras de cambio, tales como las tarjetas de pago para el uso de servicios específicos hasta llegar a las tarjetas de crédito que actualmente gozan de una amplia aceptación; la proliferación de estas planteó un problema de compatibilidad que se fue resolviendo al estandarizar los cajeros automáticos.

Algunas de las herramientas que han contribuido a la evolución del comercio electrónico, que podemos mencionar son: Internet, el intercambio electrónico de datos (EDI), seguridad en la red, publicidad y mercadeo en Internet, la computadora personal, el software, multimedia y vídeo digital, las telecomunicaciones, y la computación móvil y sin cable.

1.4.2 Definición.

El concepto de comercio electrónico o e-commerce, es de reciente creación, y se encuentra en constante evolución, es un término que se ha puesto muy de moda y puede generar confusión con el término e-business. Mientras que e-commerce se refiere a las transacciones de bienes o servicios utilizando Internet, e-business, integra Internet a todos los procesos productivos de un negocio.

Algunas definiciones de Comercio Electrónico (e-commerce) son:

"E-Commerce se refiere al proceso de comprar o vender un producto o servicio sobre una red electrónica. El medio más popular para conducir el comercio electrónico es Internet".[3]

El comercio electrónico *"se refiere generalmente a todas las formas de transacciones relacionadas con las actividades comerciales, incluyendo organizaciones e individuos, que están basadas en el proceso y transmisión de datos digitalizados, incluyendo texto, sonido e imagen"*[4]

"En términos de negocios, es una nueva metodología que ayuda a las organizaciones, proveedores de productos y/o servicios y consumidores a reducir costos mientras se mejora la calidad de los bienes y servicios y se acelera la entrega de los mismos."[5]

De estas definiciones podemos destacar el hecho de que comercio electrónico no sólo involucra a Internet, esto sería restrictivo de otros medios electrónicos; sin embargo, y para los objetivos de este trabajo nos centraremos en comercio electrónico en Internet. Podríamos decir simplemente que el comercio electrónico es cualquier operación comercial donde las partes involucradas interactúan a través de medios electrónicos. Además debemos destacar que involucra a cualquier tipo de bienes físicos tales como artículos para el hogar, u otros que

---

[3] Stephen Walther, Jonathan Levine, 2001

[4] OECD, 1997

[5] Fernández S, Manuel, Aguaita, 2001

pueden llegar al consumidor desde la misma Internet, tales como música, información, etcétera.

Al hablar de comercio electrónico debemos considerar el entorno donde se produce y los mecanismos que lo hacen posible.

1.4.3 Clasificación.

De acuerdo al tipo de bienes que se comercializan, el comercio electrónico se clasifica en:

- **Directo:** donde los bienes comercializados se hacen llegar al cliente de forma electrónica, se dice que son bienes intangibles, tales como software, sonido, etc.
- **Indirecto:** Son bienes tangibles que después de realizar un contrato se hacen llegar al cliente por los medios tradicionales de distribución, tales como muebles, libros, etc.

Otra forma de clasificar el comercio electrónico, es a través de las partes involucradas en la transacción, de esta manera tenemos:

- **Negocio a Negocio**: Conocida como B2B (Business to Business), se refiere a las transacciones realizadas entre empresas.
- **Empresa a Consumidor**: B2C (Business to Consumer), esta categoría incluye los sitios en Internet donde una empresa ofrece sus productos directamente a los consumidores.

- **Consumidor a Consumidor:** C2C (Consumer to Consumer), conocidas como subastas, donde particulares ofrecen bienes y/o servicios a otros particulares, sin mediar con empresas.

- **Gobierno – Ciudadano:** G2C (Government to Civic), no es un negocio propiamente dicho, se refiere al pago de impuestos o realización de trámites por Internet.

- **Gobierno – Negocio:** G2B (Government to Business), al igual que el anterior tampoco es un negocio, y este se refiera a los tramites entre las partes por medio de Internet.

1.4.4 Ventajas.

El comercio electrónico esta creciendo a gran velocidad, por eso es que no se duda del impacto económico y social que ha tenido y tendrá en los próximos años. Pero a pesar de ofrecernos ventajas, su inicio, no ha sido fácil, ya que se tropezó con los defensores del comercio tradicional, pero cabria destacar que el comercio electrónico no es una amenaza para este, sino un complemento.

Entre las ventajas que podemos mencionar:

Ventajas para la empresa:

- El costo transaccional es menor.
- No se necesita ser una empresa grande, para poder colocar un negocio en línea
- Reduce los costos de marketing y publicidad.
- Reduce los costos de hacer negocios.

- Incrementa su catalogo de clientes.
- Facilita los negocios Internacionales.

Ventajas para el cliente:

- Reduce la cadena de intermediarios.
- Reduce el ciclo de venta, puede tener los elementos necesarios para tomar una decisión inteligente.
- Amplia su capacidad de acceder a prácticamente a cualquier producto.
- Permite ver la competencia a un clic.

Ventajas para empresa y cliente:

- Se puede establecer una comunicación eficiente entre las partes.
- Permite realizar el proceso de venta a cualquier hora.

1.4.5 Inicio de un negocio.

Antes de poner en marcha un comercio electrónico, debemos considerar algunos puntos, para obtener éxito.

1. Iniciar con este tipo de negocios, puede ser sencillo y económico, siempre y cuando, se sepa que es lo que se debe hacer.

2. El diseño del WEB Site debe ser agradable, fácil de navegar, rápido de descargar. El nombre del dominio, debe ser fácil de escribir y sobre todo fácil de recordar para el cliente.

3. Se debe considerar la publicidad del WEB Site y las promociones en el mismo como una estrategia permanente.

4. Se debe tener definido el perfil de los posibles clientes, para que el diseño del WEB Site sea dirigido a ellos.

5. Tener soporte para la atención a clientes.

6. Diseñar estrategias para el embarque y rastreo de productos.

1.4.6 Seguridad.

El Servidor para el WEB Site debe ser seguro, la seguridad es una parte medular del comercio electrónico, tanto para clientes como para empresas, ya que el primero debe tener la seguridad de que después de pagar va a obtener el bien o servicio contratado; y la empresa debe tener la seguridad de que después de entregar el bien, o proporcionar un servicio recibirá su pago. También la seguridad involucra una de las barreras más importantes de romper por el comercio electrónico, ya que implica una nueva forma de hacer negocios, y a la cual no toda la gente esta acostumbrada o le tiene confianza.

La mayoría de las tecnologías basan su seguridad en la identificación de un usuario, a través de un nombre y una contraseña, esto puede funcionar en una red cerrada, pero cuando hablamos de Internet, se requieren sistemas más complejos, por ejemplo:

- Algoritmos cifrados, basados en el uso de llaves: algoritmos simétricos o de llave secreta, que usan la misma llave tanto para el encriptado como para el desencriptado; algoritmos asimétricos o de llave pública, usan llaves diferentes para el encriptado y desencriptado, la llave

pública se hace pública distribuyéndola libremente, mientras que la llave privada se guarda en secreto y nunca se distribuye, los datos que se cifran con la clave pública sólo se pueden descifrar con la clave privada, los datos cifrados con la clave privada sólo se descifran con la clave pública, el método de cifrado asimétrico, permite que cualquiera pueda encriptar mensajes con la llave pública, pero solo el poseedor de la llave privada podrá ver el contenido.

- Firmas digitales, una pequeña cantidad de información creada usando alguna llave secreta, y una llave pública que se utiliza para verificar que la firma fue realmente generada con la correspondiente llave privada, son utilizadas, para verificar que un mensaje realmente proviene del remitente declarado.

- Protocolo SET (Secure Electronic Transaction), se ha desarrollado con el propósito de asegurar y otorgar autenticidad a los participantes en compras abonadas con tarjeta de crédito en cualquier tipo de red en línea. Cumple con los objetivos de dar confidencialidad a la información transmitida, autentifica tanto a clientes como empresas, integra la información transmitida, permite autentificar y no rechazar por las partes las transacciones realizadas, permite operar entre distintas plataformas de hardware y software utilizados por los participantes de las transacciones.

- Protocolo SSL (Secure Sockets Layer), constituido por dos sub-protocolos: el SSL Record, que define el formato

usado en la transmisión de datos, y el protocolo SSL handshake, que involucra el primero para intercambiar mensajes entre un servidor SSL y un cliente SSL, este intercambio de mensajes esta diseñado para que el cliente autentifique al servidor, y viceversa, permite al cliente y servidor seleccionar los algoritmos criptográficos soportados por ambos y utilizar técnicas de encriptación de llave publica, para generar una llave secreta compartida, establece una conexión SSL segura.

Otro factor importante que deben considerar las empresas al conectarse a Internet, es la protección contra virus, para lograrlo el software antivirus ofrece varias opciones en cuanto a la variedad existente en el mercado, pero lo mejor es conseguir alguno que busque varias señales diferentes de virus y que sea fácil de actualizar.

1.4.7 Componentes de una Aplicación de Comercio Electrónico.

El comercio electrónico es un concepto que agrupa diferentes tecnologías para mejorar la eficiencia entre las transacciones entre usuarios, podemos mencionar:

- Intercambio electrónico de datos.
- Correo electrónico.
- Formularios y catálogos electrónicos.
- Transferencia electrónica de fondos segura.
- Correo de voz.
- Animaciones y gráficos de alta calidad.

Una vez que ya se tomo la decisión de entrar al mundo del comercio electrónico, es vital determinar el motor que hará funcionar el WEB Site, entre los componentes esenciales de una aplicación de comercio electrónico están:

- Catálogo de productos de acceso rápido, el cliente no debe esperar más de diez segundos.
- Carro de compras.
- Base de datos de productos con información detallada sobre el bien.
- Formularios para obtener información del cliente.
- Selección de forma de pago.
- Encriptamiento de la información de crédito.
- Sistema de administración de la Tienda.
- Sistema de personalización.
- Envío y rastreo de productos.
- Otros componentes no esenciales, pero que son útiles son: motores de búsqueda, estadísticas, base de datos administrativa de clientes y ordenes, capacidad para hacer promociones y descuentos.

1.4.8 Legislación sobre Comercio Electrónico en México.

En México no existe una ley como tal sobre comercio electrónico, sin embargo, los aspectos de transacciones por Internet se rigen por disposiciones incluidas en El Código Civil, El Código de Procedimientos Civiles, El Código de Comercio Federal y Ley Federal para la Protección del Consumidor que se reformaron por decreto publicado el 29 de mayo de 2000 en el Diario Oficial de la Federación.

A pesar de los avances tecnológicos y del incremento de personas que día a día tienen acceso a estos, no puede concluirse que se ha llegado a la culminación del desarrollo de las tecnologías relacionadas con la computación, por lo tanto no podemos en este momento predecir la forma de las aplicaciones futuras, ni las características que tendrán posteriormente los equipos de cómputo y la forma en que los humanos nos comunicaremos con las computadoras.

Capitulo 2:
DISEÑO DE APLICACIONES WEB.

La construcción de sistemas grandes y complejos, es normalmente un conjunto de actividades en las que se encuentran involucrados diferentes grupos de trabajo; cada uno, con actividades diferentes y bien definidas para cada etapa de la construcción del software; el diseño es una de esas etapas. Este capitulo permitirá ubicar el diseño dentro de algunos de los ciclos de vida más comunes para la construcción del software, además identificará la importancia de esta etapa y los aspectos importantes a considerar durante está, en particular para aplicaciones WEB.

2.1 El diseño dentro del ciclo de vida del software.

2.1.1 Ciclo de vida del software.

Se entiende por ciclo de vida del software al lapso de tiempo transcurrido desde que se concibe la elaboración/incorporación de un producto de software hasta el momento en que el producto es instalado y aceptado.

Durante un proyecto, la existencia del ciclo de vida del software, permite definir y estructurar las actividades a llevarse a cabo,

además proporciona puntos de control y revisión para tomar decisiones sobre la continuación del proyecto.

Existen distintas metodologías para desarrollar software, para hacer una selección apropiada para un determinado proyecto se debe considerar la naturaleza del mismo, los métodos y las herramientas a utilizarse, los controles y entregas que se requieren así como los recursos humanos y financieros disponibles.

2.1.2 Modelos de Ciclo de Vida.

Modelo de cascada.

También llamado Modelo Básico o Modelo Secuencial Lineal, es el más antiguo y el más utilizado en el desarrollo del software; sugiere una división del ciclo de vida en una secuencia de etapas bien definidas; propone planear un proyecto antes de iniciarlo, definir el comportamiento externo del sistema que se desea antes de diseñar su arquitectura interna, documentar los resultados de cada actividad, diseñar antes de construir el sistema y probarlo después de construirlo.

Normalmente enfrenta problemas, porque los proyectos raramente siguen el flujo lineal propuesto y es muy difícil para los clientes explicar todos los requisitos de un sistema al inicio del proyecto.

De acuerdo con Roger S. Pressman[1], las etapas que constituyen este modelo son:

---

[1] Pressman, Roger S. Ingeniería del Software Un enfoque práctico, 2000, p 23.

- Ingeniería y modelado de Sistemas/Información.
- Análisis de los requisitos del software.
- Diseño.
- Generación de código.
- Pruebas.
- Mantenimiento.

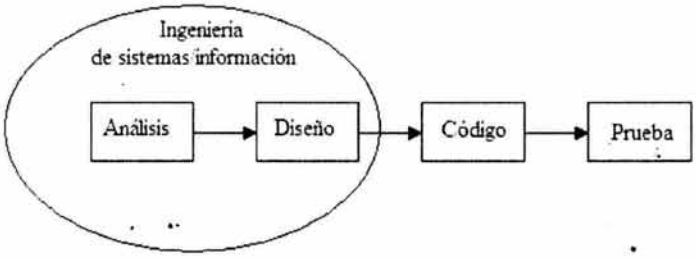

Figura 2-1: Modelo Secuencial Lineal.

Ciclo de Vida Estructurado.

Surge como un intento para comprender sistemas grandes y complejos dividiéndolo en componentes para construir un modelo del sistema; donde cada componente se transforma en un módulo independiente desde un punto de vista funcional.

Actividades del ciclo de vida estructurado[2]:

- Encuesta.
- Análisis.
- Diseño.
- Implantación.
- Generación de pruebas de aceptación.

---

[2] Yourdon, Edward, Análisis Estructurado Moderno,1993,pp 98-104.

- Garantía de calidad.
- Descripción del procedimiento.
- Conversión de base de datos.
- Instalación.

En este ciclo de vida, nada implica que una determinada actividad deba estar concluida antes de iniciar la siguiente, esto da al ciclo de vida un aspecto no secuencial.

Construcción de prototipos.

Un prototipo es una representación o modelo del producto de programación, esta alternativa consiste en definir un conjunto inicial de necesidades e implantarlas rápidamente con la intención de expandirlas y refinarlas iterativamente.

Este modelo es de utilidad cuando los requerimientos no están bien definidos, y tiene la desventaja que el cliente puede confundir el prototipo con el trabajo final y pedir que se instale. Para que funcione la construcción de prototipos el cliente como el equipo de desarrollo deben definir las reglas al inicio de un proyecto y respetarlas como mecanismo de definición de requisitos.

Fases a repetir:

- Definición de requerimientos.
- Diseño rápido de un modelo.
- Revisión del prototipo

Modelo RAD (Rapid Application Development).

Es una adaptación del modelo lineal enfatizando un ciclo de desarrollo extremadamente corto, se logra el desarrollo rápido

utilizando un enfoque de construcción basado en componentes. Es una metodología ideal para aplicaciones de mediana complejidad en un periodo extremadamente corto de desarrollo.

El modelo RAD plantea un enfoque de desarrollo iterativo, analiza rápidamente el problema del negocio, recolecta requerimientos usando grupos dirigidos, intensifica la cooperación entre usuarios y desarrolladores para diseñar una solución viable. Para conseguir la finalización de la aplicación rápidamente utiliza herramientas automáticas para la creación de prototipos y generación de código además de valerse de la reutilización de componentes del software.

Comprende las siguientes fases:

- Modelado de gestión.
- Modelado de datos.
- Modelado del proceso.
- Generación de aplicaciones.
- Pruebas y entrega.

Tiene las ventajas de reducir el tiempo de desarrollo en un 50% o más, requerir pocos recursos humanos y materiales, alto grado de participación del usuario. Las limitaciones de ese modelo son que no todos los tipos de aplicaciones son apropiados para desarrollarse con este esquema, no se puede dar al usuario al mismo tiempo velocidad de desarrollo, abatimiento de costos y calidad, además de requerir compromiso por parte de usuarios y desarrolladores.

2.1.3 Modelos de Procesos Evolutivos del Software.

Estos modelos reconocen que el software evoluciona, durante el tiempo de desarrollo los requisitos de gestión o de producto

pueden cambiar, produciendo al final un software que no es real, en general los modelos evolutivos son iterativos, permitiendo desarrollar versiones de software cada vez más completas.

Modelo Incremental.

Combina elementos del modelo secuencial, con la filosofía iterativa de la construcción de prototipos. Aplica secuencias lineales produciendo un producto operacional al final de cada secuencia, conforme progresa el tiempo en el calendario. El primer incremento es, a menudo un producto esencial que cubre requisitos básicos. Como resultado de la utilización y/o evaluación de un incremento se desarrolla un plan para el incremento siguiente; este proceso se repite hasta la elaboración de un producto completo.

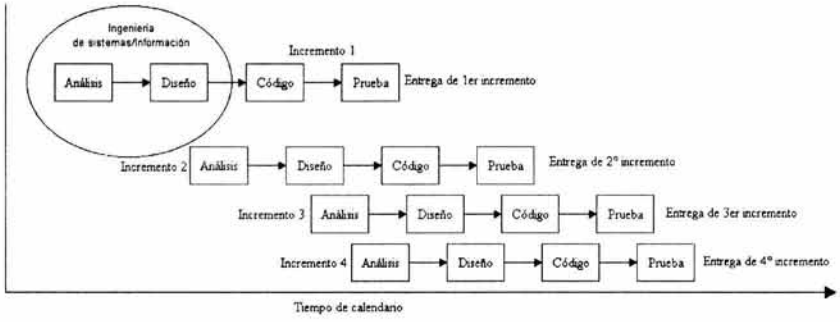

Figura 2-2: Modelo Incremental según Roger S. Pressman[3].

Tiene la ventaja de que la construcción de cada incremento es menos riesgosa que la construcción del sistema completo, es más

[3] Pressman, Roger S. Ingeniería del Software Un enfoque práctico, 2000, p 27

fácil determinar si los requerimientos han sido bien comprendidos; y es particularmente útil cuando el personal no es suficiente para una implementación completa.

Sus inconvenientes son presuponer que todos los requerimientos son definidos al inicio, además de requerir de cierta experiencia para definir los incrementos para distribuir las tareas de forma proporcional.

Desarrollo concurrente.

Es un modelo dirigido por las necesidades del usuario, las decisiones de la gestión y los resultados de las revisiones. Se puede representar como una serie de actividades técnicas importantes, definiendo una serie de acontecimientos que dispararán transiciones de estado a estado para cada una de las actividades.

Este modelo es utilizado frecuentemente para el desarrollo de aplicaciones cliente/servidor, definiendo actividades en dos dimensiones:

1. Sistemas: Diseño, ensamble y uso.

2. Componentes: Diseño y realización.

Modelo en Espiral.

Combina la naturaleza evolutiva de la construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial; proporcionando el potencial para el desarrollo rápido de versiones incrementales de software.

Generalmente en el modelo existen de tres a seis actividades estructurales o regiones de tarea:

- Comunicación con el cliente.
- Planificación.
- Análisis de Riesgos.
- Ingeniería.
- Construcción y adaptación.
- Evaluación del cliente.

En este modelo no existen fases fijas, la dirección del proyecto debe decidir como estructurarlo en fases. El paso de una fase a otra produce ajustes en el plan del proyecto. Las fases son representadas por giros en espiral con la dirección de las agujas del reloj, comenzando por el centro. El primer giro produce el desarrollo de una especificación de productos, los siguientes pasos podrán ser utilizados para el desarrollo de prototipos y progresivamente versiones más sofisticadas del software.

Para el desarrollo de sistemas y software a gran escala este modelo presenta un enfoque bastante realista, donde los desarrolladores y usuarios comprenden y reaccionan mejor ante los riesgos en cada uno de los niveles evolutivos.

Tiene la ventaja de mantener el enfoque sistemático del ciclo de vida clásico, incorporándolo a un marco de trabajo interactivo reflejando de mejor manera el mundo real.

Es un esquema relativamente nuevo en relación con los esquemas lineales o de construcción de prototipos, por ello resulta difícil convencer a los clientes de que el enfoque evolutivo es controlable y se posee con las habilidades requeridas para la evaluación del riesgo.

2.1.4 Análisis de Sistemas.

El propósito principal de las actividades del análisis es transformar las políticas del usuario y el esquema del proyecto en una especificación estructurada.

Independientemente del paradigma de desarrollo utilizado, el análisis de requisitos es la primera fase técnica del proceso de ingeniería del software; y permite descubrir, refinar, modelar y especificar en detalle el ámbito del software, creando modelos que pueden traducirse en el diseño de datos, arquitectónico, de interfaz y de procedimientos.

El análisis de requisitos del software puede dividirse en cinco áreas de esfuerzo:

1. Reconocimiento del problema.
2. Evaluación y síntesis.
3. Modelado.
4. Especificación.
5. Revisión.

Básicamente el análisis de requerimientos genera modelos de los dominios de la información, funcionales y de comportamiento.

- **Modelo de Dominio de la información:** incluye tres visiones de los datos, (1) contenido de la información y sus relaciones, (2) flujo de la información y (3) estructura de la información.

- **Modelo funcional:** se refiera a las transformaciones de la información que deberá realizar el software, para poder representar toda la funcionalidad del sistema.

- **Modelos de comportamiento:** representa la forma en que el sistema responde a los acontecimientos del mundo exterior.

El modelo del análisis más utilizado en el mundo es el análisis estructurado. El modelo debe lograr describir lo que requiere el cliente, establecer una base para la creación de un diseño del software y definir un conjunto de requisitos que se pueden validar una vez finalizado el software. Esto se logra cuando el modelo extraído durante el análisis estructurado toma la forma de la figura 2-3.

Figura 2-3: Estructura del modelo de análisis.

El centro del modelo es ocupado por el diccionario de datos que contiene definiciones de todos los objetos de datos consumidos y producidos por el software. Rodeando el núcleo se encuentran los diagramas de entidad-relación (DER), que representa las relaciones entre los objetos de datos; el diagrama de flujo de datos (DFD), que proporciona una indicación de cómo se transforman los datos y representa las funciones y subfunciones

que transforman el flujo de datos; el diagrama de transición de estados (DTE), el cual indica como se comporta el sistema como consecuencia de sucesos externos.

2.1.5 Diseño de Aplicaciones.

De forma sencilla se puede decir que un modelo es una simplificación de la realidad. El objetivo del diseño es generar modelos o representaciones de una entidad para construirla posteriormente; aplicando diversas técnicas y principios con el fin de definir un dispositivo, un proceso o un sistema con el detalle suficiente para permitir su realización física.

El diseño del software se encuentra en una fase de desarrollo temprana, y se ha tomado en serio sólo en las últimas tres décadas. Sin embargo, se encuentran disponibles, métodos de diseño, criterios de calidad de diseño y se puede aplicar una notación específica.

El diseño del software ha evolucionado desde el diseño centrado en los criterios de desarrollo de programas modulares, y métodos para refinar la arquitectura del software de una manera descendente en la jerarquía. La denominada programación estructurada es una evolución de los aspectos procedimentales de la definición del diseño; posteriormente se propusieron métodos para la transformación del flujo de datos o de la estructura de datos en una definición de diseño. Para la obtención del modelo del diseño, más recientemente, se propone un enfoque orientado a objetos.

Han surgido muchos métodos de diseño, sin embargo, todos tienen características comunes: un mecanismo para obtener una representación a partir de un modelo de análisis, una notación para

representar componentes funcionales y sus interfaces, pautas para el refinamiento y la partición y consejos para valorar la calidad.

En el nivel técnico, la ingeniería del software empieza con una serie de tareas de modelado que llevan a una especificación completa de requisitos y a una representación del diseño general del software a construir. Estas tareas permiten crear modelos que definan los procesos que satisfagan las necesidades consideradas, representen el comportamiento de los procesos, definan explícitamente las entradas de información al modelo y representen las relaciones de los modelos generados.

El modelado es la parte central de todas las actividades que conducen a la producción de buen software haciendo la diferencia entre hacer que funcione un programa y hacerlo bien; definiendo la importancia del proceso de diseño del software en una sola palabra: Calidad.

Cada uno de los elementos del modelo de análisis proporciona información necesaria para crear un modelo de diseño.

El diseño de datos utiliza los datos y las relaciones definidas en el diagrama entidad-relación así como el diccionario de datos para generar las estructuras de datos necesarias para implementar el software.

El diseño arquitectónico define del modelo de análisis y de la interacción de sus subsistemas definidos los principales elementos estructurales del programa.

El diseño de interfaz utiliza los diagramas de flujo de datos y la especificación de control para describir cómo se comunica el

software consigo mismo, con los sistemas que operan con él y con los operadores que lo emplean.

El diseño de procedimientos obtiene información de la especificación del proceso y del control y el diagrama de transición de estado para transformarla en una descripción de procedimientos de los componentes del software.

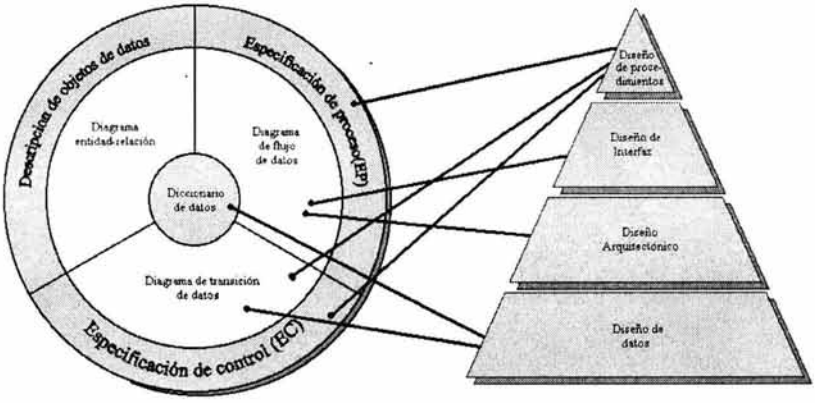

Ilustración 2-4: Transformación del modelo de análisis en un diseño del software.

Existen un conjunto de principios fundamentales y conceptos básicos al diseño de datos, arquitectónico, de interfaz y de procedimientos, que deberían aplicarse independientemente del método de diseño que se emplee.

El diseño del software es, al mismo tiempo un proceso y un modelo; como proceso es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del software a construir, en tanto que el modelo del diseño proporciona los planos del software a construir y puede involucrar planos detallados y generales, y muestra diferentes perspectivas del sistema, utilizando diferentes modelos.

Para lograr un diseño que muestre factores de calidad externos e internos[4], se deben de aplicar apropiadamente algunos principios, por ejemplo:

- El diseño debería ser uniforme: Antes de empezar a trabajar deben existir normas de estilo y de formato para el equipo de diseño.

- El diseño debería ser integrado: se deben definir las interfaces entre los componentes del diseño.

- El diseño debería estructurarse para admitir cambios y aceptar circunstancias inusuales.

- Escribir código no es diseñar, y diseñar no es escribir código.

- La calidad del diseño debería valorarse durante el proceso de diseño, no al final.

2.1.6 Conceptos del diseño.

Los conceptos de diseño proporcionan criterios básicos para la calidad del mismo. Estos conceptos ayudan a responder las siguientes preguntas:

---

[4] Los factores de calidad externos son las propiedades del software que pueden percibir los usuarios como corrección, robustez, extensibilidad, facilidad de uso, reutilización, funcionalidad y compatibilidad; y los internos son los que permiten un diseño de alta calidad desde la perspectiva técnica como modularidad o legibilidad.

- ¿Qué criterios pueden emplearse para la partición del software en componentes individuales?

- ¿Cómo se extraen la función o la estructura de datos de una representación conceptual del software?

- ¿Hay criterios uniformes que definen la calidad técnica de un diseño del software?

Abstracción.

Una abstracción es la separación de un objeto de su esencia o de sus cualidades, permite concentrarse en un problema a un nivel general independientemente de los detalles de nivel inferior.

Dentro del diseño del software se usan tres formas de abstracción:

1. Abstracción procedimental, es una secuencia dada de instrucciones con una función específica y limitada.

2. Abstracción de datos, es una colección determinada de datos que describen un objeto de datos.

3. Abstracción de control, implica un mecanismo de control del programa sin especificar detalles internos.

Refinamiento.

Es un complemento de la abstracción y ayuda a revelar detalles de bajo nivel a medida que se progresa en el diseño. Es un proceso de transformación que inicia con un enunciado con un alto nivel de abstracción y el diseñador proporciona más detalles con cada refinamiento sucesivo.

Modularidad.

Un problema complejo es más fácil de resolver cuando se divide en piezas más pequeñas y manejables. La modularidad se refiere a la capacidad que tienen los programas de dividirse en componentes con nombre y ubicación determinada, denominados módulos. Un módulo es un componente identificable del software que se puede tratar por separado y que al integrarse con otros módulos ayuda a satisfacer los requisitos del software

Arquitectura del software.

La arquitectura del software se refiere a la estructura general del software. Es la estructura jerárquica de los componentes (módulos), la forma de interactuar de éstos componentes, y la estructura de los datos usados.

El crear una versión arquitectónica del software es un objetivo del proceso del diseño, entre el conjunto de propiedades que deben especificarse en el diseño arquitectónico están las propiedades estructurales, que es la definición de los componentes del sistema; las propiedades extra-funcionales, que es la descripción de cómo la arquitectura del diseño consigue los requisitos de rendimiento, capacidad, seguridad y otras características del sistema; y familias de sistemas relacionados, el reuso de bloques de construcción arquitectónica que se encuentran comúnmente en el diseño de familias de sistemas similares.

Jerarquía de control.

También conocida como estructura del programa, y representa la organización de los componentes (módulos) del programa. El

diagrama en forma de árbol es la más común de las estructuras utilizadas para expresar las notaciones de control de los módulos.

La jerarquía de control también representa la visibilidad (scope) que muestra todos los componentes a los que un determinado módulo puede solicitar o utilizar sus datos; y la conectividad que indica, dado un determinado módulo, el conjunto de componentes a los que directamente se invoca o de los que se utilizan sus datos.

Partición estructural.

Denominada también partición arquitectónica, hace referencia a la partición del programa tanto de manera horizontal, que en su forma más simple define tres particiones entrada, transformación de datos y salida. La partición vertical, también conocida como descomposición en factores, sugiere que las funciones de control y poco trabajo de procesamiento lo realicen los módulos superiores, y los módulos ubicados en la parte baja de la arquitectura sean los módulos que realicen las tareas de entrada, cálculo y salida.

Estructura de datos.

La representación lógica existente entre los elementos individuales de datos, puede representarse con la estructura de datos. Muestra alternativas de la organización de la información; así como, métodos de acceso, capacidad de asociación y procesamiento.

Procedimiento del software.

Es una descripción del procesamiento, detallando exactamente la secuencia de eventos, definiendo los puntos de decisión y señalando operaciones repetitivas.

Ocultamiento de información.

Es la definición de restricciones de un módulo a otros externos para el acceso a procedimientos y datos propios del primero. Esto ayuda a que los errores inadvertidos no se propaguen a otros módulos.

Diseño modular.

El diseño modular coloca en cada módulo del programa una función del sistema. El principal argumento es la descomposición de un problema en partes más pequeñas y por tanto más manejables. Aporta una reducción de la complejidad, facilita los cambios, produce una implementación más sencilla y permite el desarrollo de diferentes partes de un sistema al mismo tiempo.

De acuerdo a Bertrand Meyer[5] Los criterios a aplicar para un buen diseño modular:

- **Principio de descomponibilidad modular**: Si se facilita la descomposición de un problema en varios subproblemas.

- **Principio de componibilidad modular:** Es la capacidad que tienen los módulos de combinarse con otros y esta relacionado con la reutilización de componentes.

---

[5] Programación Modular, [en línea], J.M. Ruiz M.,modificado en Ene-03, [citado el 05/abr/2003], disponible en: http://www.abacusnt.com/tem26.pdf

- **Principio de comprensibilidad modular:** Los módulos deben ser comprendidos de manera independiente.

- **Principio de continuidad modular:** Los pequeños cambios originan pequeñas repercusiones

- **Principio de protección modular:** Los errores en un módulo no deben propagarse a otros módulos.

Otro concepto básico en el diseño modular es la independencia funcional, que trata de que los módulos tengan una única función claramente definida y la interacción con otros módulos sea mínima.

La independencia funcional se puede medir con los criterios cualitativos de cohesión y acoplamiento. La cohesión es la fuerza relativa funcional de un módulo, un módulo con cohesión debería realizar una sola cosa.

El acoplamiento mide el grado de independencia que se puede obtener entre módulos que deben comunicarse.

2.2 Arquitectura de las aplicaciones.

Putman[6] define la arquitectura de una aplicación como:

"El conjunto de reglas y conceptos que definen la estructura, desarrollo semántico y relaciones entre las partes de la aplicación, incluye los elementos que la forman, las relaciones

---

[6] Putman, Janis R.,Architecting with RM-ODP,2003,disponible en http://safari.informit.com

entre estos elementos, describiendo cada elemento de la aplicación y la aplicación en su totalidad".

Se puede decir, que la arquitectura de una aplicación hace referencia a la forma en que la aplicación esta diseñada tanto lógica como físicamente. Es una vista conceptual de la estructura de la aplicación, tanto de la presentación como del procesamiento y almacenamiento de datos.

El diseño adecuado de la arquitectura de una aplicación se traduce en estructuras estables del sistema que pueden adaptarse a los cambios; asegurando así la satisfacción continúa de los requerimientos.

La arquitectura de una aplicación es quizá el elemento más importante del diseño que permite manejar diferentes puntos de vista de una aplicación, y controlar el desarrollo iterativo e incremental de una aplicación durante su ciclo de vida. Una aplicación puede describirse mejor arquitectónicamente utilizando cinco vistas interconectadas, donde cada vista es una proyección de la estructura y organización de la aplicación, centrada en un aspecto particular de la aplicación.

Vista de casos de uso: Describe el comportamiento del sistema.

Vista de diseño: Comprende los requisitos funcionales.

Vista de procesos: Cubre, principalmente el funcionamiento, capacidad, crecimiento y rendimiento de la aplicación.

Vista de Implementación: Comprende los componentes involucrados para ensamblar y hacer disponible físicamente una aplicación.

Vista de despliegue: Contienen información de la configuración del hardware sobre el que se ejecuta la aplicación.

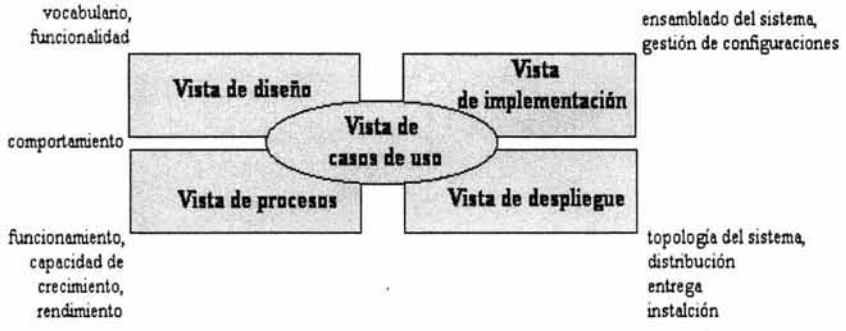

Figura 2-5: Vistas de la arquitectura de una aplicación

Cada una de las vistas puede existir por si misma, de forma tal que un usuario, analista, desarrollador, etcétera, pueda centrarse en las cuestiones de la arquitectura que más le interesen

2.2.1 Arquitectura Cliente/Servidor.

La arquitectura cliente/servidor es el resultado de la evolución de la arquitectura centralizada[7], Esta arquitectura es un modelo para el desarrollo de sistemas de información, donde los procesos se dividen en módulos independientes que cooperan entre sí para intercambiar información, servicios o recursos.

---

[7] La arquitectura centralizada es normalmente una plataforma para la programación por procedimientos, con un servidor central donde residen todos los datos y procesos de los datos, y que además controla el acceso a múltiples terminales conectadas a través de productos integrados en la arquitectura de la red.

Algunas características destacables de la arquitectura cliente/servidor son:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor.
- El cliente no depende de la ubicación física, ni del tipo de equipo físico, ni del sistema operativo del servidor.
- Los cambios en el servidor implican pocos o nulos cambios en él(los) cliente(s).

Conceptualmente los elementos de la arquitectura cliente/servidor son el cliente, el servidor y la infraestructura de comunicaciones.

2.2.2 Aplicaciones para Arquitectura Cliente/Servidor.

Las aplicaciones desarrolladas para arquitectura cliente/servidor se distinguen de otras formas de software distribuido por las siguientes características:

- La existencia de un servicio suministrado por el servidor y empleado por el cliente.
- Varios clientes pueden ser atendidos por un servidor al mismo tiempo y este regula su acceso a recursos compartidos.
- El software cliente/servidor siempre oculta al cliente la ubicación del servidor.

- El software es independiente del hardware y de las plataformas de software del sistema operativo.

- Es una infraestructura versátil, modular, basada en mensajes y tiene como objetivo mejorar la portabilidad, la interoperabilidad y la escalabilidad.

Toda aplicación de software posee tres funciones elementales: administración de datos, lógica de la aplicación (procesos) y lógica de la presentación (interfaz del usuario). La relación entre estas funciones y la arquitectura cliente servidor se encuentra presente tanto en PC's como en sistemas grandes.

Por esta razón, se derivan de la arquitectura cliente/servidor un conjunto de variantes que dependen de cuales funciones de la aplicación ejecuta el cliente y cuales el servidor.

- **Presentación distribuida:** Se distribuye la presentación entre el cliente y el servidor, y éste último almacena la totalidad de datos y ejecuta todos de procesos

- **Presentación Remota:** El cliente posee la interfaz del usuario, soportando la presentación captura y validación de datos y consultas, los procesos y los datos se encuentran en el servidor.

- **Proceso distribuido:** La presentación esta en el cliente, los datos en el servidor y la lógica de la aplicación se encuentran distribuida entre el cliente y el servidor.

- **Gestión de datos remota:** La presentación y los procesos residen en el cliente y los datos permanecen en el servidor.

- **Bases de datos distribuidas:** La presentación, los procesos y parte de los datos se encuentran en el cliente, el resto de los datos en el servidor.

Esta clasificación distribuye las partes de una aplicación en uno de los extremos de la arquitectura cliente/servidor, definiendo dos modelos básicos.

1) **Servidores amplios:** Cuando la mayor parte de la aplicación está en el servidor (presentación distribuida), son fáciles de administrar y reducen los intercambios en red mediante la creación de niveles de servicios más abstractos.

2) **Clientes grandes:** Cuando la parte preponderante de la aplicación reside en el cliente (gestión de datos remota y bases de datos distribuidas), minimiza el procesamiento en el servidor aprovechando la capacidad de procesó del cliente.

Cada modelo tiene sus propios usos, y algunas veces se complementan; así un servidor WEB es un servidor amplio, un servidor de bases de datos es un ejemplo de cliente grande y los objetos distribuidos pueden ser de ambas clases.

La clasificación de la arquitectura de aplicaciones en clientes grandes y servidores amplios no es la más utilizada, suele preferirse la clasificación en capas o niveles, aunque en esencia es lo mismo.

Aplicación de una capa.

Es la estructura más simple, se puede decir que es una sola entidad, donde la aplicación, los datos y la herramienta que maneja los datos se encuentran en la misma máquina.

Aplicación de dos capas (Two-Tier)

Sus componentes son el cliente y el servidor y los procesos se ubican con la interfaz del usuario con el cliente (Gestión de datos remota), o con la base de datos en el servidor (Presentación Remota).

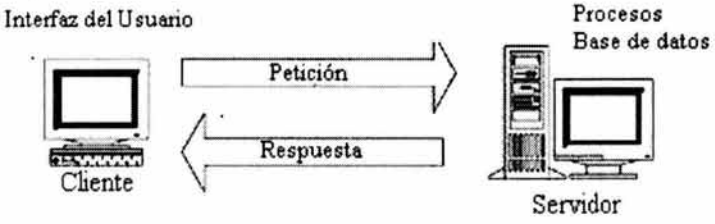

Figura 2-6: Gestión de datos remota

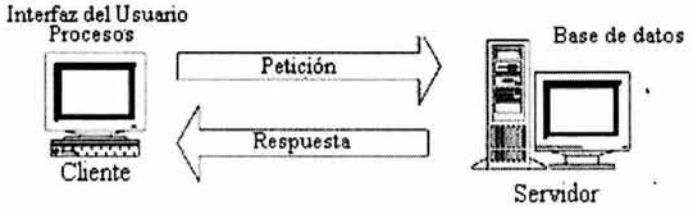

Figura 2-7: Presentación remota.

Aplicación de tres capas (Three-Tier)

Divide las funciones de la aplicación en tres componentes, Interfaz del usuario, lógica de la aplicación y la información. A este modelo se le conoce también como modelo de servicios.

1) **Interfaz del usuario:** Este componente se encarga de la interacción hombre-máquina a través del monitor, teclado, ratón o algún otro medio como el reconocedor de voz.

2) **Lógica de la aplicación:** Varios servidores o componentes de software localizados en una o más plataformas que se encargaran de conectar la interfaz del usuario y la información.

3) **Información:** Esta formada por los datos que gestiona la aplicación, que puede ser una base de datos o documentos.

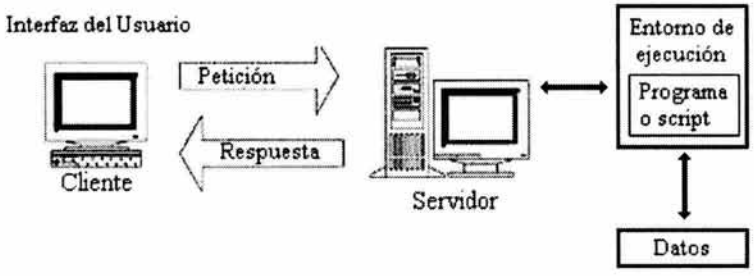

Figura 2-8: Arquitectura de tres capas.

2.3 Consideraciones para el diseño de aplicaciones WEB.

En el diseño de una aplicación WEB, la mayoría de las actividades son iguales que para cualquier sistema cliente/servidor: se reparten objetos en los módulos del sistema y se desarrollan las clases necesarias de la infraestructura para agregar al modelo del análisis.

El repartir apropiadamente los objetos del negocio en una aplicación WEB es crítico y depende de la arquitectura de la aplicación. Los objetos pueden residir en el servidor, el cliente, o en ambos. En una aplicación sofisticada, los objetos de validación de entradas de datos funcionarán probablemente en el cliente, mientras que los objetos como una lista de clientes o el

catálogo de productos, existirán solamente en el servidor. Algunos objetos, tales como una factura, pueden tener vida en ambos.

Los elementos del diseño de una aplicación WEB se descubren primero observando el modelo de experiencia del usuario y entendiendo el documento de la arquitectura del software, que delinea las reglas para la creación de los elementos del diseño de la aplicación WEB y las funciones de cada elemento.

2.3.1 Modelo UX (Experiencia del Usuario)

El término de experiencia del usuario (UX) está recibiendo una atención especial para el desarrollo de aplicaciones WEB y es responsable de crear lo que se ve y se siente en la aplicación, determinando las rutas de navegación principales a través de las páginas WEB del sistema, y manejando/organizando la estructura del contenido en las páginas.

Los tres aspectos claves del modelo UX son:

1) **Las páginas y su contenido:** en este sentido una página es algo que se presenta al usuario, que adicional a la infraestructura propia (menús, controles, etcétera), contienen información que es relevante para el usuario, y que puede ser de contenido estático o dinámico.

2) **Los escenarios:** que son expresiones específicas del uso del sistema, están compuestos de páginas combinadas, que expresan pequeñas historias de la aplicación.

3) **Las rutas de navegación:** es el más importante de los artefactos que proporciona el modelo, y provee información

sobre las páginas de la aplicación con sus posibles rutas de navegación generando un mapa con ellas y definiendo la legalidad de estas rutas dentro del sistema.

Para el desarrollo del modelo UX la creación de prototipos es la actividad más importante, los prototipos pueden tomar muchas formas y son buenos para crear secuencias de eventos que describen situaciones específicas al navegar por las pantallas del sistema. En las aplicaciones WEB, la interfaz del usuario es casi siempre un sistema de páginas WEB con contenidos estáticos y dinámicos, esta naturaleza dual de las aplicaciones indica que el diseño WEB debe identificar los elementos dentro de la aplicación y definir su naturaleza.

Así podemos identificar que en el paso del tiempo, durante la fase de diseño tendremos un diseño de aplicación como una interfaz y otro como un sistema de hipertexto, el primero esta orientado a tareas y el segundo a información.

Sin importar la existencia de herramientas que faciliten la labor del diseño, el diseñador siempre tendrá que tomar algunas decisiones que puedan ser la diferencia entre un buen diseño y uno malo. Algunas pautas para tomar esas decisiones:

1) La tecnología avanza rápidamente, y es difícil conocer cuales de las nuevas características podrán convertirse en un estándar, por lo tanto se debe ser cuidadoso con las nuevas capacidades de la tecnología.

2) Se debe intentar mantener baja la complejidad del lado del cliente, dado que en esta parte el equipo de desarrollo tiene menos control.

3) Planear primero, después crear el modelo, ya que el modelo es la base del desarrollo del software de calidad, no se debe apresurar la fase de diseño; y para iniciar el diseño es necesario tener claro que se va a construir. Responder preguntas del tipo ¿Quién va a usar la aplicación?, ¿Qué se va a lograr al usar la aplicación?, ¿Qué información será requerida? Y ¿Cuáles son las características de los usuarios?

4) Si el usuario requiere llenar un formulario, antes se le deberá proporcionar cierta información: ¿qué puede lograr?, ¿qué información se requiere?, ¿cómo será utilizada la información?, ¿cuánto tiempo llevará completar la forma? y ¿qué puede esperar el usuario al terminar?.

5) El objetivo de una forma es recoger los datos necesarios de forma rápida y fácil. Para lograr esto, y hacerlo de forma eficiente se debe considerar lo siguiente: agrupar los elementos por tema, determinar si es necesaria una sola página o páginas múltiples, que sea fácil para los usuarios. Con relación a este último punto es importante para los usuarios saber exactamente que se quiere, evitar rescribir cuando sea posible, validar datos del lado del cliente.

6) La disposición visual en una forma, puede hacer más fácil o más difícil que el usuario llene el formulario. Para hacerlo más fácil, hay que ajustar los formularios a un formato para ser funcional y visualmente atractivo.

7) Por supuesto, es muy importante validar datos en los formularios, para conseguir los datos que se solicitan y en el formato correcto, para evitar la mecanografía de

errores se pueden pre-llenar las respuestas fiables, evitar los mensajes secretos.

8) Antes que un usuario termine de llenar una forma, se le debería permitir hacer cambios, y una vez concluido el proceso debería recibir una confirmación de que la información fue recibida.

Con respecto a la apariencia de las páginas que conforman una aplicación WEB, deben considerarse los siguientes puntos:

Color.

Lo primero que se observa al visitar una página, es el color, deben utilizarse pocos colores y se debe producir una sensación de unidad en todas las páginas de la aplicación. También es conveniente elegir un color de fondo y texto que contrasten para hacer fácil la lectura.

Navegación:

Los elementos que permiten al usuario moverse dentro de una página ó hacia otras páginas deben ser fácilmente identificados.

Contenido.

Es lo más importante, pero debe buscarse que los textos sean breves, y que las líneas de los textos no ocupen toda la pantalla o la excedan, deben incluirse tablas siempre que esto facilite la lectura.

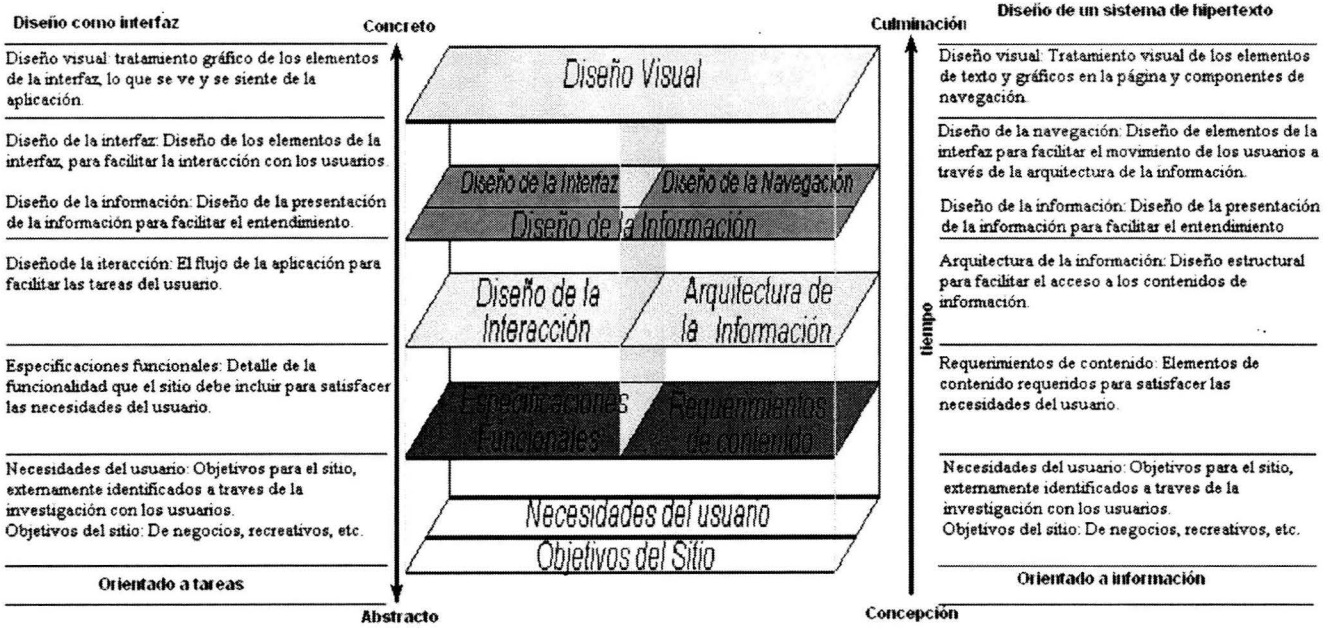

Figura 2-9: Elementos del modelo de la Experiencia del Usuario (Modelo UX).

Imágenes y multimedia.

Deben incluirse solo las imágenes, sonidos y animaciones necesarias para evitar la lentitud de la aplicación.

De esta forma hemos identificado el diseño como una etapa importante y clave en el éxito de un proyecto de software, es una fase común a diferentes modelos de ciclo de vida. El diseño tendrá éxito si puede expresarle al usuario lo que éste espera obtener en el producto de software una vez que este concluido, en particular para aplicaciones WEB, se deben atender las recomendaciones mencionadas en cuanto al diseño de pantallas y su secuencia para obtener un producto que satisfaga a sus clientes.

Capitulo 3:
EXTENSIÓN WAE PARA UML.

La etapa del diseño, al igual que el análisis y el modelo UX, producen resultados que deben ser evaluados tanto por el equipo de trabajo, como por el usuario del software, para poder tener éxito en la comunicación, es importante manejar un lenguaje común para ambos participantes, el Lenguaje Unificado de Modelado (UML) es una herramienta que pretende proveer de ese lenguaje común. Este capítulo se enfoca a conocer algunas de las características de UML y en particular su extensión de notación WAE y la forma en que estas herramientas pueden ser aprovechadas durante estas fases de la construcción de aplicaciones de software, en particular para aplicaciones WEB.

3.1 UML: Lenguaje Unificado de Modelado

3.1.1 Definición de UML.

El Lenguaje Unificado de Modelado (UML) es un lenguaje estándar que hacer modelos que nos permitan construir software, proporciona un vocabulario y un conjunto de reglas para utilizarlo posibilitando la representación física y conceptual de un sistema.

El vocabulario y las reglas de UML indican como crear y leer modelos bien formados, proporcionando una herramienta para

visualizar las vistas de un sistema mientras evoluciona a través del ciclo de vida del software.

Un proceso bien definido para el desarrollo del software, guiara a los usuarios en la toma de decisiones acerca de los artefactos a producir, las actividades y el personal empleado para crearlos y gestionarlos, y como usar los artefactos para medir y controlar el proyecto de forma global. UML no nos dice que modelos crear y cuando, esta es una tarea del proceso de desarrollo del software.

3.1.2 Historia de UML

La notación UML se deriva de la unificación de las tres metodologías de análisis y diseño orientado a objetos más destacadas:

1. Metodología de Grady Booch para la descripción de conjuntos de objetos y sus relaciones, de manera particular durante la fase de diseño y construcción de proyectos.

2. La Técnica de modelado orientada a objetos de James Rumbaugh (OMT: Object-Modeling Technique), útil principalmente para el análisis y en sistemas con gran cantidad de datos.

3. Ingeniería del Software Orientada a Objetos de Ivar Jacobson (OOSE: Object- Oriented Software Engineering), que soporta los casos de uso como forma de dirigir la captura de requisitos, el análisis y el diseño de alto nivel.

Los principales objetivos al inicio de la unificación de las metodologías fueron[1]:

1. Modelar sistemas, desde el concepto hasta los artefactos ejecutables, utilizando técnicas orientadas a objetos.

2. Cubrir las cuestiones relacionadas con el tamaño, inherentes a los sistemas complejos y críticos.

3. Crear un lenguaje de modelado utilizable tanto por las personas, como, por las máquinas.

El esfuerzo para unificar estas metodologías se inicio en 1994, cuando Rumbaugh se unió a Booch, con un proyecto cuyo objetivo principal fue la unificación del método Booch y OMT, creando el borrador de la versión 0.8 del entonces denominado Método Unificado y el cual fue publicado en octubre de 1995. En esos días Jacobson se integro al equipo y el alcance del proyecto se amplio para incorporar OOSE. Para junio de 1996, los esfuerzos del equipo de condujeron a la versión 0.9. Para entonces, ya muchas organizaciones de software veían a UML como un punto estratégico para su negocio y se estableció un consorcio de UML con varias organizaciones que querían dedicar recursos para trabajar una definición fuerte y completa para UML. Entonces se produjo UML 1.0 el cual se ofreció para su estandarización al Object Management Group (OMG) en enero de 1997. y el 14 de noviembre de 1997 la versión 1.1 de UML fue adoptada por el OMG.

---

[1] Booch, Grady. Rumbaugh, James. Jacobson, Ivar. El lenguaje Unificado de Modelado, pp XXII y XXIII

En relativamente poco tiempo UML se ha convertido en el lenguaje de modelado dominante del mercado. UML puede considerarse como un estándar, independientemente del paradigma de desarrollo utilizado, de hecho se consideran fuera del ámbito de UML tanto los lenguajes de programación, como las herramientas, como el ciclo de vida del software.

3.1.3 Aspectos y usos de UML.

Básicamente UML puede utilizarse para visualizar, especificar, construir y documentar los aspectos estructurales (estáticos) y de comportamiento (dinámicos) de un sistema que involucra una gran cantidad de software.[2]

Para visualizar:

Cuando se escriben modelos en UML, la creación de modelos explícitos facilitan la comunicación entre las personas involucradas en un proyecto, ya que UML incluye una semántica bien definida para cada símbolo de la notación UML, así algún desarrollador puede escribir un modelo UML y otro desarrollador, o incluso otra herramienta, puede leer el modelo sin ambigüedad.

UML es un lenguaje gráfico que facilita la comprensión de ciertas estructuras que son más fáciles de comprender en modelos que trasciendan los lenguajes de programación textual.

---

[2] Booch, Grady. Rumbaugh, James. Jacobson, Ivar. El lenguaje Unificado de Modelado, pp 11

Para especificar:

UML cubre la especificación de todas las decisiones de análisis, diseño e implementación que deben realizarse al desarrollar y desplegar un sistema con gran cantidad de software, construyendo modelos precisos, no ambiguos y completos.

Para construir:

UML no es un lenguaje de programación visual, sin embargo, sus modelos pueden conectarse de forma directa a una gran variedad de lenguajes de programación; definiendo una correspondencia de un modelo UML a un lenguaje de programación como Java, C++ o Visual Basic. Además UML es lo bastante explícito como para permitir la ejecución directa de modelos, la simulación de sistemas y la instrumentación de sistemas en ejecución.

Para documentar:

UML cubre la documentación de la arquitectura de un sistema incluyendo todos sus detalles, también proporciona un lenguaje para expresar requisitos y pruebas; y para modelar las actividades de planificación de proyectos y gestión de versiones.

UML esta pensado principalmente para sistemas con gran cantidad de software, y puede ser usado efectivamente en dominios tales como:

- Sistemas de información de empresas.
- Bancos y servicios financieros.
- Telecomunicaciones.
- Transporte.

- Defensa/industria aerospacial.
- Comercio.
- Electrónica médica.
- Ámbito científico.
- Servicios distribuidos basados en la WEB.

3.1.4 EL Modelo UML.

Los tres elementos principales de UML son los bloques de construcción, las reglas semánticas y mecanismos comunes, para interpretar el modelo conceptual del lenguaje es necesario comprender estos componentes.

Los bloques de construcción se combinan para la creación de modelos bien formados, utilizando las reglas para hacerlos semánticamente consistentes y coherentes con todos los modelos relacionados.

Los mecanismos comunes de UML se aplican de forma consistente para crear modelos simples y armoniosos que se ajusten a un patrón de características comunes.

Al avanzar en el conocimiento de UML los modelos que se obtiene son cada vez más completos, integrando cada vez más elementos del vocabulario, de está forma se avanza de un modelado estructural básico hasta el modelado avanzado del comportamiento.

3.1.5 Elementos estructurales.

Los elementos estructurales son en su mayoría las partes estáticas de un modelo, y representan cosas materiales o conceptuales.

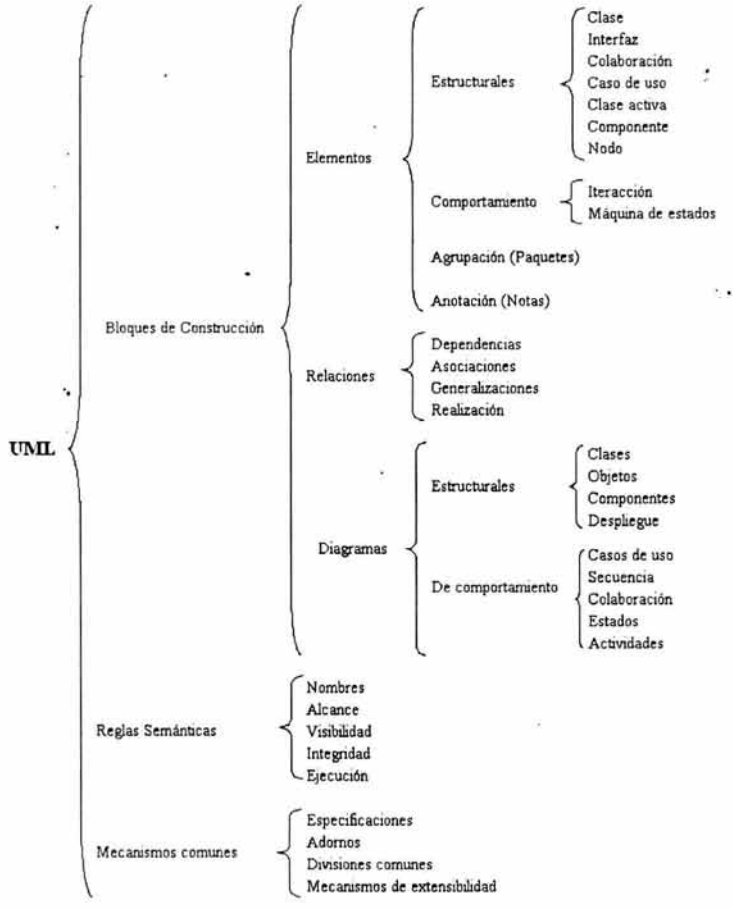

Figura 3-1: Modelo Conceptual de UML.

Clase.

Las clases son bloques de construcción fundamentales en cualquier sistema orientado a objetos, pueden utilizarse para representar software, hardware, o cosas sencillamente conceptuales, describiendo un conjunto de objetos con atributos, operaciones, relaciones y semántica en común. Una clase es una

abstracción de las cosas que forman parte del vocabulario del sistema que sé esta desarrollando, es una representación de un conjunto de objetos y no un objeto individual.

Cada clase debe tener un nombre único dentro de un paquete[3], el nombre simple de una clase se forma por cualquier número de letras, números y ciertos signos de puntuación (excepto los dos puntos, que se reservan para el nombre de camino o ruta, que indica el nombre simple de la clase y el del paquete que lo contiene), en la práctica los nombres de clase son cortos, y normalmente la primera letra de cada palabra se pone en mayúscula.

Gráficamente la representación gráfica de UML para una clase en un rectángulo y puede dibujarse mostrando únicamente el nombre de la clase.

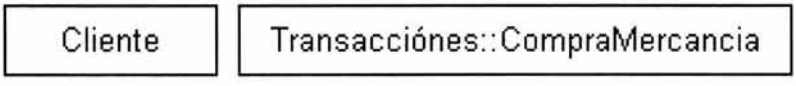

Figura 3-2: Nombre simple y nombre de camino para una clase.

Un **atributo** de una clase es la identificación por medio de un nombre de una característica del objeto que se modela. Una clase puede tener cualquier cantidad de atributos, inclusive cero; gráficamente los atributos se listan debajo de la sección del nombre de la clase, los atributos pueden representarse mostrando solo sus nombres, indicando el tipo de dato del atributo, e inclusive mostrando un valor inicial.

---

[3] Un paquete es un mecanismo de propósito general para organizar elementos en grupos.

El nombre de un atributo esta formado por texto, igual que el nombre de la clase, y se usan mayúsculas para la primera letra de cada palabra dentro del nombre del atributo, excepto la primera letra.

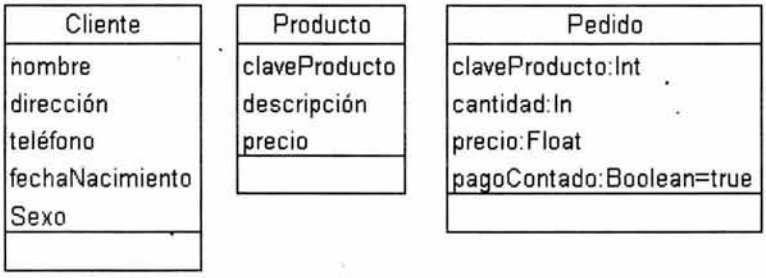

Figura 3-3: Atributos de una clase.

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, una clase puede tener o no operaciones, con frecuencia cuando se llama a una operación sobre un objeto, los datos o el estado de éste, pueden variar, el nombre de una operación puede ser texto, donde la inicial de cada palabra, excepto la primera son mayúsculas, y suelen ser verbos cortos.

Gráficamente se representan en la sección que se encuentra debajo de los atributos, y se pueden declarar indicando su signatura, es decir su nombre, tipo y valores por defecto de todos los parámetros, y el tipo de retorno para el caso de las funciones.

Con frecuencia, una clase posee demasiados atributos y operaciones, lo que complica listarlos todos cuando se dibuja una clase, de hecho del total, solo un subconjunto será relevante para una vista especifica, por ello puede decidirse mostrar solo algunos o ninguno de sus atributos y operaciones; cuando una

lista finaliza con puntos suspensivos, indica que existen más operaciones y/o atributos.

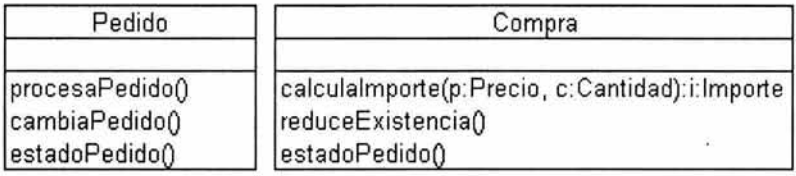

Figura 3-4: Operaciones de una clase.

Una **responsabilidad** es una obligación de una clase, y los atributos y operaciones de la clase tienen la finalidad de satisfacerlas, en la práctica, una clase tiene por lo menos una responsabilidad. Gráficamente se expresan por medio de texto libre (limitado a un párrafo corto), en una sección al final de la representación de la clase. Una clase ayuda a identificar "cosas" dentro del problema o de la solución.

Interfaces.

UML, utiliza las interfaces para separar la implementación de una clase de su especificación, son conjuntos de operaciones que se utilizan para detallar un servicio de una clase o un componente.

Una interfaz debe tener un nombre único, que lo distinga de otras interfaces dentro del paquete al que pertenece, y puede estar formado por cualquier cantidad de letras, números y algunos signos de puntuación exceptuando los dos puntos que se reservan para la separación del nombre de camino, para distinguir una interfaz de una clase se precede el nombre de la clase con una "I". Gráficamente una interfaz es representada en forma normal como un círculo y puede incluir solamente su nombre, omitiendo

la visualización de las operaciones que colecciona, cuando es importante para comprender un modelo particular, se puede presentar una interfaz como una clase estereotipada, listando sus operaciones en la sección apropiada.

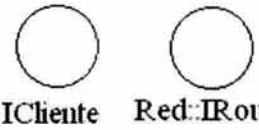

Figura 3-5: Representación de una interfaz.

Ilustración 3-6: Operaciones de una interfaz.

Además una interfaz puede proporcionar información sobre otras propiedades como visibilidad, alcance y semántica de concurrencia.

Las clases implementan a una o más interfaces, es decir realizan las interfaces, si una interfaz es considerada un contrato para una clase, la clase esta obligada a cumplirlo.

Para hacer más comprensible y manejable una interfaz, se pueden:

1. Asignar pre y postcondiciones a cada operación.
2. Usar OCL[4] para especificar formalmente la semántica.

---

[4] Lenguaje de Restricciones de Objetos (OCL, Object Constraint Language) de UML, que permite especificar la semántica de forma precisa describiendo restricciones de manera fácil y evitando que sean ambigüas.

3. Asociar máquinas de estados a las interfaces
4. Emplear colaboraciones para detallar el comportamiento esperado de la interfaz.

Un **tipo,** es un estereotipo de una interfaz, y se utiliza para especificar un dominio de objetos, para distinguirlo de una clase o una interfaz, se precede el nombre de una "T", como en TCaracter.

Un **rol** es el comportamiento de una entidad participante en un contexto dado.

Colaboración.

Es una sociedad de clases, interfaces y otros elementos que participan para proporcionar un comportamiento cooperativo mayor que la suma de comportamientos de sus elementos. Se utilizan para especificar la realización de casos de uso y operaciones. Desde el punto de vista de la arquitectura del sistema permite modelar mecanismos significativos que incluyen tanto aspectos estáticos como dinámicos del sistema.

Gráficamente una colaboración se representa con una elipse con el borde discontinuo. Debe tener un nombre único que la distinga de otras colaboraciones dentro del paquete que la contenga, este debe estar formado por letras, números y signos (a excepción de los dos puntos que separan el nombre del nombre del paquete que la incluye), normalmente la primera letra del nombre es mayúscula. Una colaboración sólo debe representarse explícitamente cuando sea necesario comprender sus relaciones con otros elementos del sistema global, de otra manera, las colaboraciones deben mantenerse en la especificación del sistema.

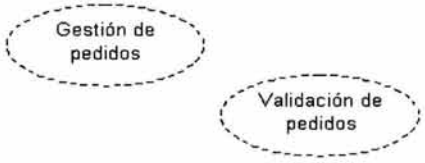

Figura 3-7: Colaboraciones.

El primer aspecto de una colaboración, se refiere a una parte **estructural,** que especifica las clases, interfaces y otros elementos que participan para llevar a cabo la colaboración, esta parte, normalmente se representa por medio de un diagrama de clases.

La parte de **comportamiento** de una colaboración se representa por medio de uno o más diagramas de interacción y deben ser consistentes con sus partes estructurales, es decir que los objetos que aparecen en las interacciones de una colaboración deben ser instancias de clases de la parte estructural.

Caso de uso.

Es la especificación de un conjunto de acciones secuénciales y sus variantes, que un sistema, subsistema, clase o interfaz realiza para producir un resultado observable. Se emplean para describir el comportamiento deseado, sin especificar como se obtiene ese comportamiento.

Cada secuencia representa una interacción de los elementos externos del sistema con el propio sistema. "Estos comportamientos son funciones al nivel del sistema que se utilizan durante la captura de requisitos y el análisis para visualizar, especificar, construir y documentar el

comportamiento esperado del sistema. Un caso de uso representa un requisito funcional del sistema"[5].

Gráficamente UML permite representar los casos de uso como elipses y los actores como monigotes. Cada caso de uso debe tener un nombre único que lo distinga de otros casos de uso dentro del paquete que lo contiene, esta formado por letras, números y signos de puntuación, exceptuando los dos puntos que separan el nombre simple del nombre del paquete en que incluye.

Figura 3-8: Caso de uso y actores.

Un **actor** normalmente es un rol que es jugado por una persona, un dispositivo de hardware o incluso otro sistema al interactuar con el sistema que se modela. Representa elementos externos, se utilizan en los modelos del sistema, pero no forman parte del mismo. Los actores se conectan a los casos de uso por medio de asociaciones, indicando que el actor y el caso de uso se comunican entre sí.

---

[5] Booch, Grady. Rumbaugh, James. Jacobson, Ivar. El lenguaje Unificado de Modelado, pp 192.

Clase Activa.

Al igual que un componente y un nodo, son elementos similares a una clase, en tanto que describen conjuntos de objetos que comparten los mismos atributos, operaciones, relaciones y semántica; pero cada uno de ellos presenta características importantes que hacen relevante un tratamiento individual en el modelado.

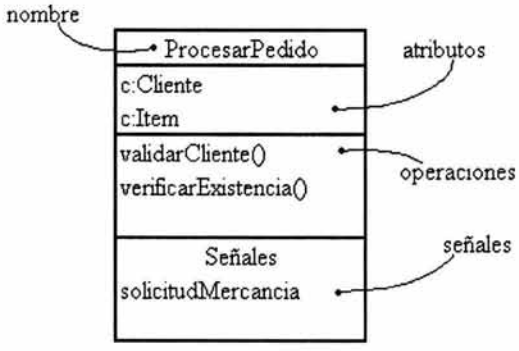

Figura 3-9: Clase activa.

Las instancias de una clase activa son objetos activos, es decir objetos que tienen procesos o hilos[6] y pueden iniciar actividad de

[6] Un **proceso** es un flujo pesado que se puede ejecutar concurrentemente con otros procesos. Un **hilo** es un flujo ligero y se puede ejecutar concurrentemente con otros hilos dentro del mismo proceso. El flujo (de control) se refiere a la secuencia con que se realizan las actividades dentro de un sistema, así en un programa secuencial se observa un solo flujo, en tanto que en un sistema concurrente, existe más de un flujo, es decir puede ocurrir más de una cosa a la vez. La distinción entre proceso e hilo surge de las dos formas diferentes en que el sistema operativo del nodo en que reside el objeto que se modela puede gestionar un flujo de control.

control. Al ser la clase activa un tipo de clase, la representación gráfica que proporciona UML para esta, es similar a la clase pero con líneas más gruesas, y tiene normalmente las mismas secciones. Adicionalmente una clase activa recibe con frecuencia señales, las cuales se enumeran en un compartimiento adicional.

Una clase activa facilita el modelado de la comunicación entre elementos activos del mismo nivel.

Componentes.

Es una parte física y reemplazable de un sistema que raramente se encuentra aislado, se utilizan para modelar los elementos físicos que pueden hallarse en un nodo, por ejemplo: ejecutables, bibliotecas, archivos y documentos. UML permite representar directamente componentes COM+, bibliotecas de código objeto, ejecutables y Enterprise Java Beans. Un componente es la implementación física de otros elementos tales como clases y colaboraciones, donde las interfaces son el puente entre los componentes y las clases.

Se dice que la interfaz es de exportación, si es realizada por un componente que presta sus servicios a otros componentes; y de importación si es utilizada por el componente.

Gráficamente se representa como un rectángulo con pestañas, debe tener un nombre que lo distinga del resto de los componentes, el nombre puede estar formado por cualquier cantidad de letras, número y signos de puntuación a excepción de los dos puntos, además puede incluir extensiones como java y dll, de acuerdo al sistema operativo destino.

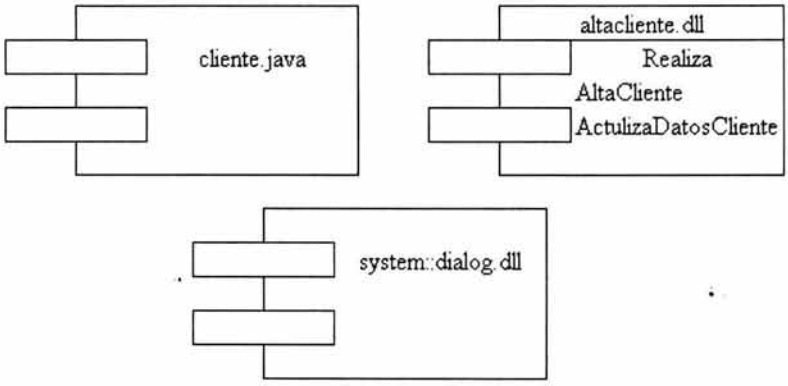

Figura 3-10: Componentes.

Básicamente existen tres tipos de componentes.

1. **De despliegue:** Son los componentes necesarios y suficientes para formar un sistema ejecutable, bibliotecas dinámicas y ejecutables.
2. **Producto del trabajo:** Productos que quedan al final del proceso de desarrollo, código fuente y archivo de datos.
3. **De ejecución:** Se crean como consecuencia de sistema de ejecución, como un objeto COM+.

Nodos.

Representa un recurso físico que existe al momento de ejecutar la aplicación, representa algún tipo de recurso de computadoras que normalmente posee memoria y con frecuencia capacidad de procesamiento.

Cada nodo debe tener un nombre que lo distinga de otros nodos dentro del paquete que lo contiene; debe ser texto formado por letras, números y signos de puntuación diferentes a los dos puntos.

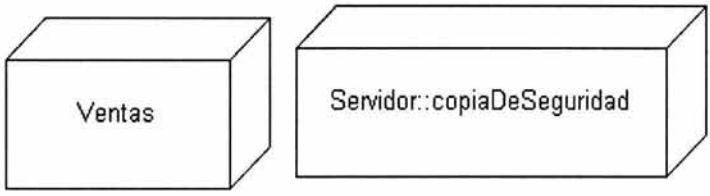

Figura 3-11: Nodos.

3.1.6 Elementos de comportamiento.

Dentro de los modelos UML, los elementos de comportamiento son las partes dinámicas, y representan el comportamiento en el tiempo y en el espacio. Los principales elementos de comportamiento son las interacciones y las máquinas de estados.

Interacciones.

Es de esperarse que dentro de cualquier sistema, los objetos no se encuentren aislados, más bien están enlazados unos a otros, así una interacción es un procedimiento que comprende un conjunto de mensajes que se intercambian entre objetos dentro de un contexto para conseguir un objetivo.

Se utilizan para modelar el flujo de control dentro de una operación, una clase, un componente, un caso de uso o un sistema; son como algoritmos bien estructurados: eficientes, sencillos, adaptables y comprensibles.

En una interacción pueden aparecer objetos como elementos concretos que representan algo del mundo real, o bien como elementos prototipo que representan instancias de los objetos.

A la conexión semántica entre objetos se les conoce como **enlace**, y especifica un camino a lo largo del cual un objeto puede enviar un mensaje a otro objeto (o a sí mismo). Los estereotipos estándar que permiten especificar la visibilidad de un objeto son association, self, global, local, y parameter.

Un **mensaje** es la especificación de una comunicación entre objetos, donde se transmite información con la posibilidad de desencadenar una actividad, que es una instrucción ejecutable que constituye una abstracción de un procedimiento computacional, y que además pueda producir un cambio de estado.

Tipos de acciones que pueden modelarse en UML:

- **Llamada**: Invoca una operación sobre un objeto.
- **Retorno:** Devuelve un valor al invocador.
- **Envío:** Envía una señal a un objeto.
- **Creación:** Crea un objeto.
- **Destrucción:** Destruye un objeto.

La representación gráfica de un mensaje es una línea dirigida, que incluye casi siempre el nombre de su operación.

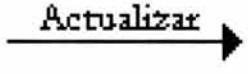

Figura 3-12: Mensaje.

Cuando se modelan interacciones con varios flujos de control involucrados, es importante distinguirlos, lo cual permite UML, escribiendo el número de secuencia de un mensaje precedido del nombre del proceso o del hilo que es el origen de la secuencia.

Máquina de estados.

Especifica los estados por los que un objeto o una interacción pasa a lo largo de su vida en respuesta a eventos, pueden utilizarse para modelar el comportamiento de cualquier elemento de modelado, pero generalmente para clases, casos de uso o sistemas completos.

Un **estado** es una situación o condición durante la vida de un objeto, durante la cual satisface una condición, realiza una actividad o espera algún evento; un estado tiene un nombre , que es una cadena de texto que lo distingue de otros estados. Acciones de entrada y salida, transiciones internas, subestados y estados diferidos. Gráficamente se representa con un rectángulo con las esquinas redondeadas.

Existen dos estados especiales, el inicial, que indica el punto de comienzo por defecto de la máquina de estados, gráficamente es un círculo negro; el estado final, que se representa con un círculo negro dentro de uno blanco y que indica que la ejecución de la máquina de estados o del estado que lo contiene ha concluido.

Adicionalmente, los estados pueden clasificarse en simples, si no poseen una subsetructura, y compuestos si tienen subestados o estados anidados, los cuales son útiles para modelar comportamientos complejos.

Un **evento (de disparo)** es la especificación de un acontecimiento significativo, que para el contexto de las máquinas de estados, es la aparición de un estímulo que puede desencadenar una transición de estados.

Una **transición** es una relación entre dos estados, indica que un objeto que está en el primer estado (estado origen) realizara ciertas acciones y entrará en el segundo estado (estado destino).

Una **condición de guarda** es una expresión booleana que se evalúa una vez por cada transición, cuando ocurre un evento.

Una **acción** es una computación atómica ejecutable, es decir no puede ser interrumpida por un evento, por lo cual es ejecutada hasta su conclusión,

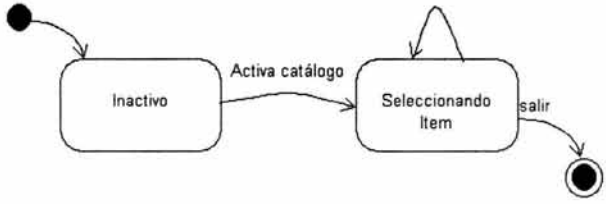

Figura 3-13:Máquina de estados.

3.1.7 Elementos de agrupación.

Los elementos de agrupación permiten descomponer un modelo en grupos organizados. Solamente existe un elemento de agrupación: los Paquetes.

Paquetes.

Los paquetes permiten organizar los elementos de un modelo en partes mayores que se pueden manipular. Gráficamente se representa como una carpeta, cada paquete debe contener un nombre que lo distinga de otros paquetes y esta formado por letras, números y signos a excepción de los dos puntos.

Los paquetes pueden incluir clases, interfaces, componentes, nodos, colaboraciones, casos de uso, diagramas e incluso otros paquetes.

Los paquetes son un mecanismo importante para que los modelos se enfrenten al crecimiento y no se creen modelos con una enorme cantidad de elementos con nombres únicos, ya que los paquetes pueden contener otros paquetes, los modelos se pueden descomponer jerárquicamente.

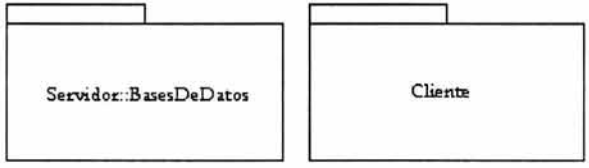

Figura 3-14: Paquetes.

La **visibilidad** de un elemento indica si este puede o no ser utilizado por otros elementos, Para especificar la visibilidad de un elemento contenido en un paquete, se antepone al nombre del elemento el símbolo de visibilidad apropiado ('+' para elementos públicos, '#' para elementos protegidos y '-' para elementos privados).

3.1.8 Elementos de anotación.

Los elementos de anotación permiten explicar los modelos UML, son comentarios que describen, clarifican y permiten hacer observaciones o restricciones sobre cualquier elemento de un modelo. Gráficamente una nota se representa como un rectángulo con la esquina doblada y puede contener cualquier combinación de texto y gráficos. El contenido de una nota no altera el significado del modelo al que se anexa.

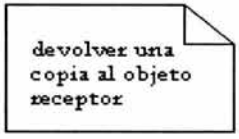

Figura 3-15: Nota.

3.1.9 Relaciones

Una relación es un vínculo entre elementos. Las relaciones más importantes son la dependencia, la asociación, la generalización y la realización. En general una relación se representa como una línea, y dependiendo de su tipo será la forma de la línea.

Dependencia.

Es una relación semántica entre dos elementos, en la cual un cambio en un elemento denominado independiente afecta la semántica del otro elemento denominado dependiente. Gráficamente se representa como una línea discontinua dirigida hacia el elemento del cual se depende; puede tener nombre, pero es raro que lo necesite, a menos que se tenga un modelo con muchas referencias.

Mayoritariamente las dependencias se usan en el contexto de las clases para indicar que una clase utiliza a otra como argumento en la signatura de una operación.

Figura 3-16: Dependencia.

Generalización.

La generalización en ocasiones se denominada una relación "*es un tipo de*", en la cual un elemento específico conocido como subclase o hijo es un tipo de elemento más general también conocido como padre o superclase. En el modelado las generalizaciones son frecuentes entre clases e interfaces para reflejar relaciones de **herencia** donde una clase hereda las propiedades (atributos y operaciones) de la clase padre.

Gráficamente la generalización se representa como una línea dirigida con una punta de flecha vacía apuntando al padre; puede tener nombre, pero, a menos que exista un modelo con muchas generalizaciones que requieran identificarse, se necesitará.

Se dice clase raíz o base a la clase sin padres y uno o más hijos, y clase hoja a la clase sin hijos.

Figura 3-17: Generalización.

Asociación.

La asociación es una relación estructural que especifica la conexión entre los objetos de dos o más elemento; la cual puede conectarse a un extremo de la relación a otro sin importar el orden.

Gráficamente y de forma básica se representa por una línea continua que conecta la misma o diferentes clases. Además se aplican los adornos de nombre, rol, multiplicidad y agregación.

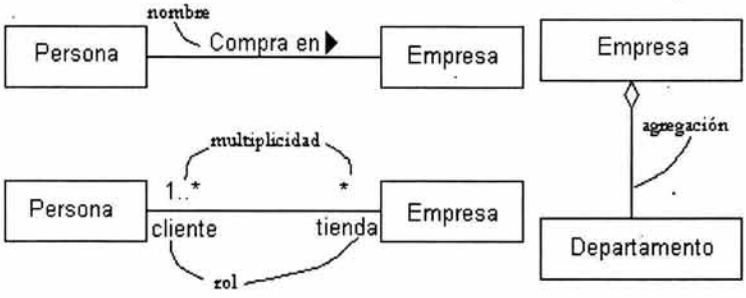

Figura 3-18: Asociaciones y sus adornos.

Una asociación puede o no tener **nombre**, este se utiliza para describir la naturaleza de la relación y para evitar ambigüedad en su significado se le puede asignar dirección al nombre por medio de una flecha que apunte en la dirección que debe ser leído el nombre.

El **rol** es la vista que presenta un extremo de la asociación al otro extremo.

Al número de objetos que pueden conectarse a través de una instancia de una asociación se le denomina **multiplicidad** del rol de la asociación.

Cuando se desea modelar algo grande (considerado un todo) que está formado por elementos (partes) más pequeños se emplea una **agregación** para indicar esta relación "tiene un". Gráficamente se le representa añadiendo un rombo vacío en la parte del todo a una asociación.

Realización.

La realización es una relación que tiene sentido entre clasificadores[7], en la cual un clasificador establece un contrato que el otro cumplirá. Gráficamente su notación es una mezcla de dependencia y generalización con una línea continua dirigida con una punta de flecha vacía que apunta al clasificador que especifica el contrato.

Esta relación es utilizada en: el contexto de las interfaces, para especificar la relación entre una interfaz y una clase o componente que proporciona una operación o servicio para ella; y en el contexto de las colaboraciones, para especificar la relación entre un caso de uso y la colaboración que realiza ese caso de uso.

3.1.10 Reglas de UML.

Los elementos de construcción y de relación no deben combinarse de forma arbitraría, como cualquier lenguaje UML proporciona un conjunto de reglas que especifican como hacerlo, para obtener modelos bien formados, semánticamente coherentes con los modelos relacionados y con el mismo. Las reglas semánticas en UML determinan el significado de cada elemento dentro de un diagrama.

---

[7] En UML un clasificador es un elemento de modelado que puede tener instancias, como por ejemplo las clases.

Regla semántica de:	Indican:
Nombres:	Como llamar a los elementos, relaciones y diagramas.
Alcance	El contexto que da un significado específico a un nombre.
Visibilidad	Como pueden ser vistos y utilizados los nombres por otros.
Integridad	Como se relacionan apropiada y consistentemente unos elementos con otros.

Tabla 3-1: Reglas semánticas de UML.

3.1.11 Mecanismos comunes.

UML provee las herramientas necesarias para visualizar, especificar, construir y documentar los artefactos de una gama muy amplia de sistemas con gran cantidad de software, sin embargo en algunas circunstancias es necesario salir de los límites de la notación proporcionada para extenderla o modificarla. UML proporciona cuatro mecanismos comunes para aplicar consistentemente dentro del modelado: especificaciones, adornos, divisiones comunes y mecanismos de extensibilidad.

Especificaciones.

A cada elemento con una representación gráfica en UML corresponde una especificación que proporciona una explicación

de la sintaxis y semántica del elemento; de esta forma se utiliza la notación gráfica para visualizar un sistema y las especificaciones para enunciar los detalles del sistema.

Adornos.

Los adornos se utilizan para mostrar detalles de la especificación de un elemento, complementando gráfica o textualmente la notación del mismo. El adorno más importante es la nota.

Divisiones comunes.

Durante el modelado de sistemas orientados a objetos se encuentra frecuentemente la división entre abstracciones y sus instancias y la separación entre interfaz e implementación.

Abstracciones y sus instancias.

Una abstracción es el conjunto de características esenciales que distinguen a cada entidad, y una instancia es una manifestación concreta de una abstracción. Las instancias no aparecen aisladas, normalmente se encuentran relacionadas a una abstracción, presentando una relación abstracción/instancia, dentro de UML la más común es clase/objeto donde el objeto es una instancia de una clase; pero se pueden presentar en casos de uso e instancias de casos de uso; componentes e instancias de componentes; nodos e instancias de nodos; y asociaciones e instancias de asociaciones.

Gráficamente una instancia se representa con el mismo símbolo de la abstracción, pero subrayando el nombre, el nombre que

debe existir y ser único dentro del contexto en el que el objeto existe (operación, componente o nodo)

Además de ocupar un espacio en el mundo real, un objeto es algo sobre lo que se pueden ejecutar operaciones y cambiar su estado.

Interfaz e implementación.

Mientras que una interfaz declara un contrato, la implementación es una realización concreta de ese contrato. Este tipo de relación puede verse en casos de uso y las colaboraciones que los realizan, así como las operaciones y los métodos que los implementan.

Mecanismos de extensibilidad

UML no es un lenguaje cerrado, por el contrario, ofrece mecanismos de extensibilidad como los estereotipos, valores etiquetados y restricciones, que de aplicarse de forma controlada, hacen posible la extensión y configuración del lenguaje proporcionando elementos suficientes para adaptarse a diferentes proyectos de software.

Estereotipos.

UML puede extenderse por medio de estereotipos, permitiendo crear nuevos tipos de bloques de construcción específicos a un problema y que derivan de los bloques existentes. A diferencia de una clase padre en una relación de generalización padre/hijo, por que cuando se aplica un estereotipo a cualquier bloque de construcción se crea un nuevo bloque de construcción como cualquiera de los existentes, pero con sus propias características

(su propio conjunto de valores etiquetados), semántica (restricciones) y notación (icono) especiales.

Un estereotipo se presenta como un nombre entre comillas tipográficas, por ejemplo: «nombre», y se coloca sobre el nombre de otro elemento; se puede, además, definir un icono para el estereotipo y colocarlo a la derecha del nombre como una señal visual.

Valores etiquetados.

Todo elemento de UML posee un conjunto de propiedades, por ejemplo las clases tienen nombres, atributos y operaciones, así como se pueden añadir nuevos elementos a UML con los estereotipos, los valores etiquetados permiten agregar nuevas propiedades a los elementos. Las etiquetas pueden definirse para los elementos existentes o para aplicarse a estereotipos individuales; a diferencia de un atributo, el valor etiquetado aplica al propio elemento y no a las instancias del elemento.

Un valor etiquetado puede verse debajo del nombre de otro elemento, como una cadena de caracteres entre llaves {}.

Restricciones.

Todos los elementos de UML poseen una semántica propia, las restricciones permiten añadir nuevas reglas o modificar las existentes. Estas pueden ser escritas en texto libre, o bien usando OCL.

3.1.12 Diagramas.

UML permite ver diferentes perspectivas de un sistema de software por medio de representaciones gráficas denominados

diagramas, en este contexto, las vistas más importantes son de casos de uso, diseño, procesos, implementación y despliegue; en cada vista se modelan cosas estáticas y cosas dinámicas. Normalmente un diagrama se muestra como un grafo conexo de nodos (elementos) y arcos (relaciones).

La representación más común de los elementos y relaciones de UML se define en nueve diagramas básicos, que se clasifican en estructurales y de comportamiento, los primeros definen las partes estáticas del modelo; los segundos se utilizan para modelar las partes dinámicas del sistema. Para proyectos específicos pueden definirse nuevos diagramas propios del proyecto.

Los diagramas en UML se utilizan básicamente para especificar modelos a partir de los cuales se creara un sistema ejecutable, y reconstruir modelos a partir de partes de sistemas ejecutables.

Sugerencias para obtener buenos diagramas.

- Decidir qué vistas se necesitan para expresar mejor la arquitectura del sistema, así como los artefactos que se necesitan para capturar los detalles esenciales.
  - Recordar que el propósito de los diagramas es implantar un sistema ejecutable, y no dibujar imágenes bonitas.
  - No complicar el diseño con diagramas extraños o redundantes.
  - Resaltar solo el detalle suficiente en cada diagrama, y evitar niveles de precisión muy alto, a menos que en realidad se requieran.
  - Mantener el equilibrio entre diagramas estructurales y de comportamiento.

- Evitar diagramas muy grandes (difíciles de navegar) y diagramas muy pequeños (triviales).
- Agrupar los diagramas en paquetes de acuerdo a la vista que se modela.
- El nombre debe utilizarse para expresar el objetivo del diagrama.

3.1.13 Diagramas Estructurales.

Los diagramas estructurales en UML son cuatro, se utilizan para especificar las partes estáticas de un sistema, y se organizan en general, alrededor de los principales grupos de elementos que aparecen al modelar un sistema.

1) Diagrama de Clases.

Son los diagramas más utilizados en el modelado orientado a objetos, contiene clases, interfaces, colaboraciones; relaciones de dependencia, generalización y asociación, además puede contener notas, restricciones, paquetes y subsistemas.

Son útiles para modelar la vista estática de un sistema, identificando los requerimientos funcionales, es decir los servicios que el sistema debe proporcionar a sus usuarios finales, estos diagramas son usados para modelar el vocabulario del sistema, colaboraciones simples o un esquema lógico de base de datos.

Permiten modelar el vocabulario identificar que esta dentro y que esta fuera del sistema en consideración, así como modelar las abstracciones correspondientes y sus responsabilidades.

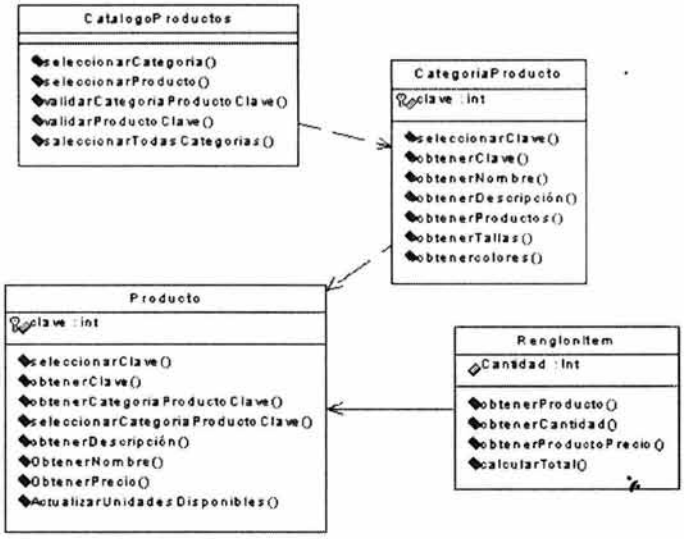

Figura 3-19: Diagrama de clases.

2) Diagrama de Objetos.

Los diagramas de objetos modelan las instancias de los elementos contenidos en los diagramas de clases, representando un conjunto de objetos y sus relaciones en un momento determinado. En este contexto se define un objeto como una cosa (no solo puede referirse a elementos físicos del sistema, sino, también a la representación de un concepto del sistema) con la que se puede interactuar, enviándole mensajes a los cuales reacciona.

El diagrama de objetos posee las propiedades comunes de los demás diagramas, y se distingue por su particular contenido de objetos y enlaces, representados gráficamente como una colección de nodos y arcos.

Figura 3-20: Diagrama de objetos.

3) Diagrama de Componentes.

Cuando se modelan los aspectos físicos de un sistema a construir, existen dos diagramas principales: de componentes y de despliegue, también denominados diagramas de implementación.

El diagrama de componentes muestra los enlaces de comunicación entre elementos de hardware, como máquinas y otros recursos como impresoras; además las relaciones entre máquinas y procesos, es decir, qué se ejecuta y en dónde.

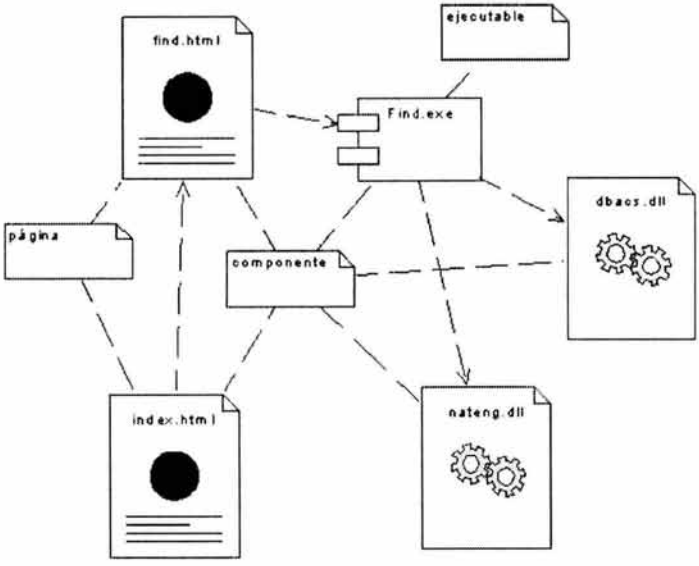

Figura 3-21: Diagrama de componentes.

Normalmente un diagrama de componentes es un tipo especial de plano que se usa para modelar la vista de implementación estática de un sistema, regularmente incluye componentes, interfaces, relaciones, y puede contener paquetes y subsistemas además de notas y restricciones. Al modelar la vista de implementación estática de un sistema se pueden utilizar los diagramas de componentes para modelar código fuente, versiones ejecutables, bases de datos físicas o sistemas adaptables.

4) Diagrama de Despliegue.

El otro diagrama utilizado para modelar los aspectos físicos de un sistema orientado a objetos es el diagrama de despliegue, muestra la configuración de los nodos que participan en la ejecución y los componentes que residen en ellos.

El contenido común de los diagramas de despliegue son nodos, relaciones de dependencia y asociación, pueden contener componentes, los cuales deben residir en algún nodo, también pueden incluir paquetes y subsistemas, además de notas y restricciones.

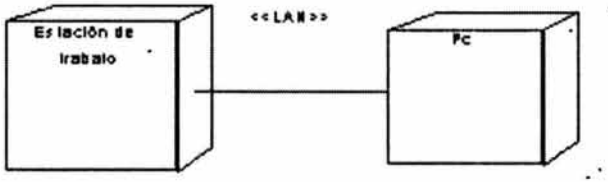

Figura 3-22: Diagrama de componentes.

Para sistemas pequeños, que residen en un solo equipo, es innecesario desarrollar diagramas de despliegue, pero, para sistemas no triviales la vista de despliegue estática de un sistema cubre la distribución, entrega e instalación de las partes que

integran el sistema físico. Los diagramas de despliegue se utilizan para modelar sistemas empotrados, sistemas cliente/servidor y sistemas completamente distribuidos.

3.1.14 Diagramas de comportamiento.

Los diagramas de comportamiento básicos de UML son cinco, se emplean para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema; involucrando cosas como el flujo de mensajes entre elementos a lo largo del tiempo y el movimiento físico de componentes en una red.

Dentro de los diagramas de comportamiento se encuentran los diagramas de casos de uso, los diagramas de secuencia, diagramas de colaboración, los diagramas de actividades y los diagramas de estados. Los diagramas de secuencia y de colaboración se conocen también como diagramas de interacción.

Los diagramas de interacción permiten mostrar en detalle cómo interactúan objetos para ejecutar una tarea, lo que implica modelar instancias de clases, interfaces, componentes y nodos, junto con los mensajes que son enviados entre ellos ilustrando su comportamiento dentro del contexto de un escenario. Normalmente contienen objetos, enlaces y mensajes; y pueden contener notas y restricciones.

Los diagramas de secuencia y colaboración son semánticamente equivalentes, lo que significa que son producidos por la información, y que se puede partir de uno de ellos, para obtener el otro, sin perder información; dada esta propiedad se dice que son diagramas isomorfos.

1) Diagrama de Casos de uso.

Estos diagramas son útiles para modelar el comportamiento de un elemento en un sistema, subsistema o clase. Presentan una vista externa de cómo puede utilizarse un elemento en un contexto dado, mostrando un conjunto de casos de uso, actores y sus relaciones. Debe poseer un nombre y su contenido es una proyección de un modelo, normalmente contiene casos de uso, actores y relaciones de dependencia, generalización y asociación; además puede contener paquetes, instancias de casos de uso; notas y restricciones.

El comportamiento del sistema puede ser documentado desde el punto de vista de un usuario utilizando los diagramas de casos de uso, son relativamente fáciles de comprender de forma intuitiva, inclusive si se desconoce la notación, ayudan a modelar el contexto de un sistema y los requisitos del mismo.

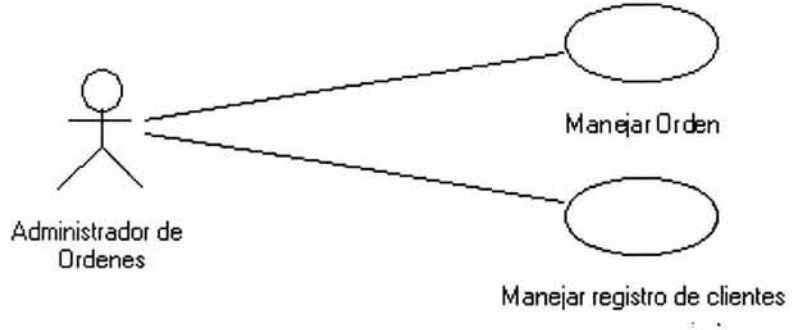

Figura 3-23: Diagrama de casos de uso.

2) Diagrama de Secuencia.

Es un diagrama de interacción que resalta la ordenación temporal de los mensajes. Gráficamente es una tabla, que representa

objetos, colocados a lo largo del eje X, y mensajes ordenados a lo largo del eje Y, según suceden en el tiempo.

Dentro de los diagramas de secuencia se pueden destacar dos características:

1. La **línea de vida** de un objeto, como una línea discontinua vertical que representa la vida de un objeto por un periodo de tiempo.

2. El **foco de control,** que indica el período de tiempo durante el cual un objeto ejecuta una acción, y se representa con un rectángulo delgado y estrecho.

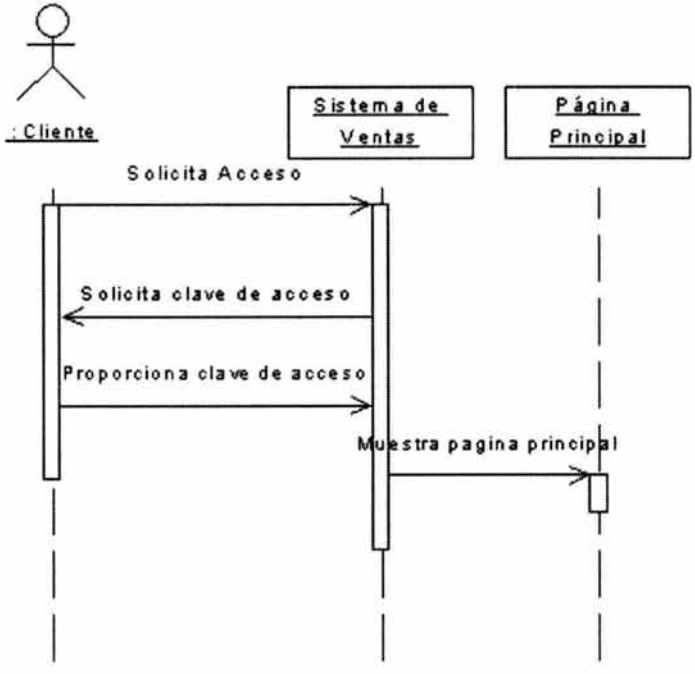

Figura 3-24: Diagrama de secuencia.

3) Diagrama de Colaboración.

Es un diagrama de interacción que destaca la organización de los objetos que participan en una interacción. Gráficamente es una colección de nodos y arcos.

Dentro de los diagramas de colaboración se pueden destacar dos características:

1. El **camino**, que indica cómo se enlaza un objeto a otro.

2. El **número de secuencia** que indica la ordenación temporal de un mensaje.

Figura 3-25: Diagrama de colaboración.

4) Diagrama de Estados.

Los diagramas de estados se utilizan para modelar aspectos dinámicos de un sistema, mostrando máquinas de estados, y destacando el flujo de control entre los estados. Gráficamente se muestran como una colección de nodos y arcos, y

fundamentalmente contiene: estados simples y compuestos; transiciones, incluyendo eventos y acciones; además puede contener notas y restricciones.

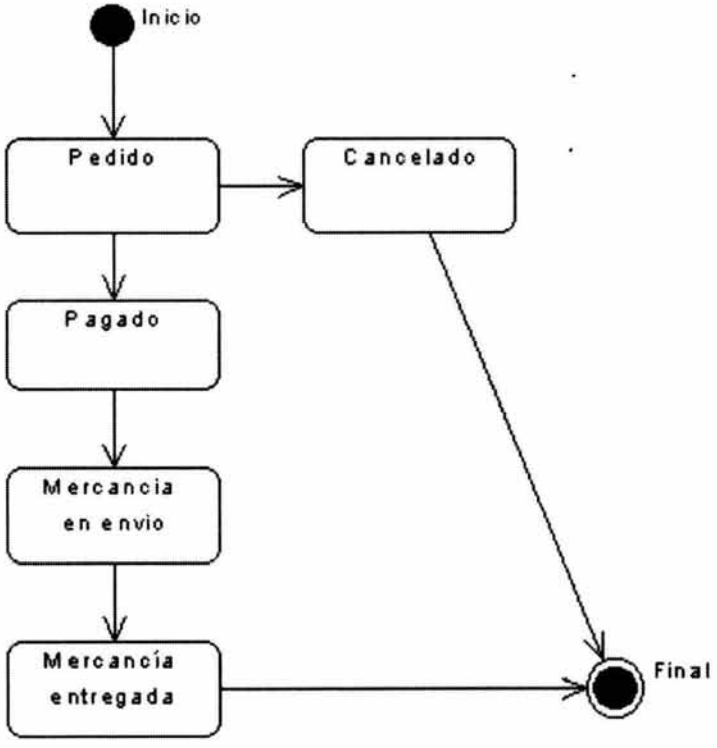

Figura 3-26: Diagrama de estados para un pedido.

La mayoría de las veces crear diagramas de estados implica modelar el comportamiento de objetos reactivos o dirigidos por eventos, es decir, objetos cuyo comportamiento se caracteriza mejor señalando su respuesta a los eventos externos a su contexto, además los objetos dirigidos por eventos son objetos con ciclos de vida bien definidos y cuyo comportamiento se ve afectado por su pasado y habitualmente su estado es de reposo,

en espera de algún evento que le demande una respuesta, y posterior a ella, regresará a su estado de reposo.

Los diagramas de estados, son también útiles para modelar la vida de un objeto reactivo, desde su creación y hasta su destrucción, identificando sus estados estables, es decir aquellos en los que el objeto puede satisfacer una condición durante un periodo de tiempo identificable; y que además pueda implicar un cambio de estado posterior a la ocurrencia de un evento.

5) Diagrama de Actividades.

Fundamentalmente el diagrama de actividades muestra el flujo de control entre actividades, donde una actividad puede ser una llamada a otra operación, envío de señales, creación o destrucción de objetos o simples cálculos que devuelván un valor o producen un cambio en el estado del sistema. Gráficamente un diagrama de actividades es una colección de nodos y arcos.

Normalmente contiene estados de actividad y de acción además de transiciones y objetos y puede contener restricciones.

Los **estados de actividad** pueden descomponerse y representar su actividad por medio de otro diagrama de actividad, en cambio un **estado de acción** no puede descomponerse ni tampoco, interrumpirse.

Una **transición** es una línea dirigida que muestra el cambio de un estado de actividad o acción a uno siguiente, además de la transición secuencial existe la **bifurcación** que se utiliza para especificar una ruta alternativa, dependiendo del valor obtenido en una expresión booleana, adicionalmente, por medio de una

línea vertical u horizontal ancha, puede representarse una **barra de sincronización** para especificar la unión y división de flujos concurrentes.

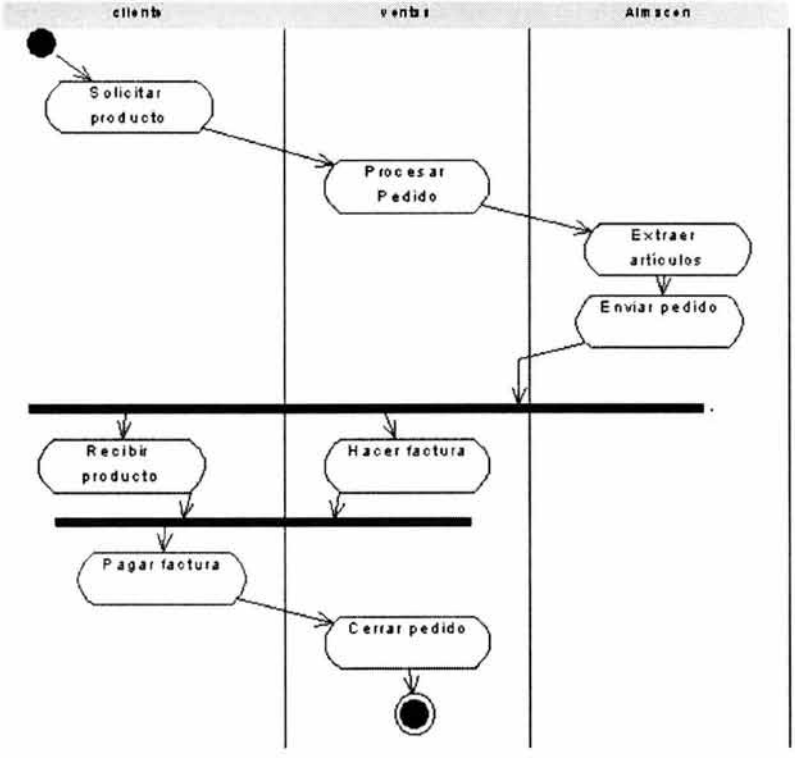

Figura 3-27: Diagrama de actividades.

3.2 Extensión WAE para UML

Una de los aspectos más sobresalientes de UML es su capacidad para absorber nueva semántica sin alterar su lógica interna. Modelar y especificar aplicaciones WEB con arquitecturas complejas con nodos dispersos geográficamente y múltiples capas de componentes, normalmente representa un reto. La extensión de la notación UML denominada **WAE (WEB**

Application Extension) desarrollada en 1998 por Jim Conallen permite aprovechar toda la gramática interna de UML para modelar aplicaciones con elementos específicos de la arquitectura de un entorno WEB.

Esta extensión amplia la notación de UML con los estereotipos, valores etiquetados y las restricciones necesarias para modelar los elementos de la arquitectura especifica de aplicaciones WEB, permitiendo que los componentes particulares de estas aplicaciones puedan ser representados en el mismo modelo y con los mismos diagramas que describen al resto del sistema en cuestión.

Actualmente en Internet el principal lenguaje utilizado es HTML, que unido al protocolo HTTP forman una estructura básica dentro de la arquitectura para aplicaciones WEB. La extensión WAE describe de forma equilibrada una asignación de elementos de UML a elementos de HTML y viceversa. Además varios estereotipos de la extensión pueden ser aplicados a las páginas WEB, que son otro elemento significativo en la arquitectura de las aplicaciones WEB.

Los objetivos principales de esta extensión son:

- Modelar los artefactos apropiados de las aplicaciones WEB, por ejemplo: Páginas WEB, relaciones entre las páginas, rutas de navegación, scripts del lado del cliente, y generación de páginas del lado del servidor.

- Modelar una aplicación WEB en el nivel apropiado de abstracción y de detalle.

- Permitir que los elementos específicos de una aplicación WEB puedan interactuar durante el modelado con el resto de los elementos de un sistema.

Al final del proceso de diseño se debe obtener un modelo UML que exprese la ejecución de la lógica de un negocio dentro del sistema. En el modelo se ejecuta parte de la lógica del negocio por los objetos del lado del servidor y los componentes en la capa intermedia de una arquitectura tres capas, y algunos otros por los elementos tradicionales del WEB, por ejemplo browsers y scripts del lado del cliente.

3.2.1 Vista lógica con la extensión WAE para UML.

En UML la vista lógica de un modelo esta compuesta por clases, sus relaciones y sus colaboraciones, las clases estereotipadas Server Page, Client Page y HTML Form permiten dentro de esta vista modelar la arquitectura básica de una aplicación WEB basada en páginas WEB. Las relaciones básicas entre estos elementos son los estereotipos de asociación: «link», «build», «submit», «redirect», «forward», «object» e «include».

«Server Page».

Es un estereotipo definido a partir del elemento de construcción clase, representa a una página WEB dinámica que ejecuta scripts en el servidor cada vez que son requeridos, interactuando con los recursos del lado del servidor tales como bases de datos, lógica de la aplicación, componentes, sistemas externos, entre otros. Las operaciones del objeto y sus atributos representan las funciones es el script y las variables que son visibles en el ámbito de la página.

Las relaciones de los Server Page están limitadas con los objetos pertenecientes al servidor. Gráficamente se representa cómo una hoja con la esquina derecha doblada y la esquina izquierda sobre una rueda dentada.

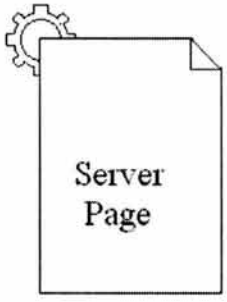

Figura 3-28: Icono Server Page.

«Client Page»

Este estereotipo es una clase y sus instancias son páginas WEB con formato HTML que incluyen datos, presentación e incluso lógica. Son representadas por los navegadores clientes y pueden contener scripts que el navegador interpreta, no tiene restricciones, y puede tener asociaciones con otras páginas cliente o servidor.

Valores etiquetados:

EtiquetaTitulo, el título que muestra el navegador de la página.

EtiquetaBase, el URL base por referencia de URLs relativos.

EtiquetaCuerpo, conjunto de atributos para la ctiqueta <cuerpo> que establecen el color de fondo y los atributos por defecto.

Figura 3-29: Icono Client Page.

«HTML Form.»

La clase estereotipada como «form» es parte de una página cliente y es una colección de campos de entrada, los atributos de esta clase representan los campos de entrada del formulario HTML (input boxes, text areas, radio buttons, check boxes, y campos hidden), no contiene operaciones, y no posee restricciones.

Los valores etiquetados GET y POST, son los métodos que se utilizan para obtener datos y proporcionarlos a la página actual.

Figura 3-30: HTML Form

Estereotipos de asociación.

«link» Es una asociación en un diagrama de clases entre una página cliente y un recurso del lado del servidor, el cual puede pertenecer a la clase «client page»o «server page», los valores etiquetados que utiliza son la lista de nombres de parámetros que deberían ser pasados con la petición de la página enlazada.

«build» Las páginas servidor existen únicamente en el servidor y son utilizadas para crear páginas cliente, la asociación «build» determina la responsabilidad de una página servidor en la creación de una determinas página cliente, identificando a cada ejecución de la pagina servidor una salida de HTML.

«submit» Es una relación direccional que se presenta siempre entre un formulario «HTML form» y una página servidor «Server Page», en la cual los formularios envían información contenida en sus campos al servidor a través de las páginas servidor, los valores etiquetados que utiliza son la lista de nombres de parámetros que deberían ser pasados con la petición de la página enlazada.

«redirect» Es una relación entre páginas cliente o servidor con otra pagina, si se origina desde una página de servidor determina que el procesado de la página solicitada debe continuar en la otra página indicando que la página destino siempre participa en la creación de la página cliente, y si se origina desde un cliente indica que la página destino será automáticamente solicitada por el

navegador. La cantidad de tiempo que una página cliente debe esperar para ser redirigida a la siguiente página es el valor etiquetado conocido como demora.

«forward» Es una relación entre una página servidor y una página cliente o página servidor, representa que el proceso de una solicitud por parte de un cliente el pedido de un cliente se delega a un recurso del lado del servidor.

«object» Una relación que dibuja un compartimiento para una clase lógica dentro de una página cliente, típicamente representa un applet, un control activeX, o cualquier otro componente incrustado.

«include» Una asociación direccional de una página de servidor a otra página de servidor o página cliente. Durante el montaje de la página en el tiempo de ejecución, esta asociación indica que la página incluida es procesada, si es dinámica, y que su contenido o subproductos son utilizados por la página padre.

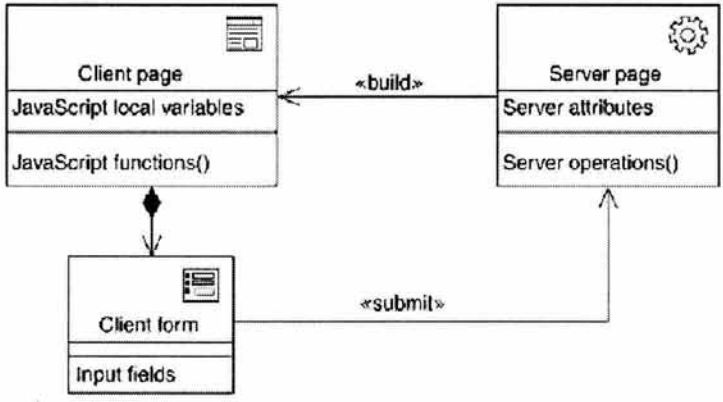

Figura 3-31: Relaciones básicas entre estereotipos de la extensión WAE.

La figura muestra las relaciones fundamentales entre las clases estereotipadas, donde todas las clases son implementadas con un componente simple, con un archivo ASP (Active Server Page) o JSP (JavaServer Page). La clase «Server Page» modela la estructura de las páginas en el servidor. Cuando una página es requerida, el servidor WEB llama el archivo correspondiente a la creación del componente requerido y ejecuta sus funciones del lado del servidor, lo cual produce una salida normalmente codificada en HTML[8] y que es aceptado y reconocido por el navegador del cliente.

3.2.2 Vista física con la extensión WAE para UML.

Los diagramas de componentes en UML se refieren a los archivos distribuidos que forman un sistema. La extensión WAE proporciona dos estereotipos de componentes, los cuales son esencialmente abstractos, puesto que se espera que sean substituidos por los componentes estereotipados específicos de un lenguaje (por ejemplo: "JSP", "XML", etcétera). Estos componentes tendrían restricciones adicionales particulares a su lenguaje y ambiente. Los dos estereotipos «static page» y «dinamic page» son suficientes para capturar el nivel de abstracción necesario para entender las características arquitectónicas significativas de las propiedades de la página componente.

---

[8] La salida puede producirse en cualquier otro formato como XML, WML(Wireless Markup Language), o imágenes, sonido, etcétera

«static page»

Es un estereotipo del elemento componente, y representa un recurso que puede ser solicitado directamente por el navegador de un cliente. Una página estática no realiza ningún proceso del lado del servidor, esto quiere decir que están restringidos y no pueden realizar componentes lógicos que sean ejecutados en el servidor, solo pueden realizar páginas cliente.

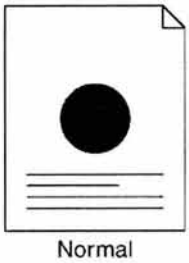

Figura 3-32: Icono «static page».

«Dynamic page»

Se trata de un estereotipo abstracto, es un recurso que puede ser solicitado por un cliente y requiere delegar el proceso del lado del servidor, el resultado de este proceso puede cambiar el estado del servidor y ser utilizado para construir el contenido dinámico de la página que responda a la solicitud del cliente. Las páginas dinámicas pueden aceptar entrada de datos por medio de los formularios. Tiene que realizar una sola página servidor, y no utiliza ningún valor etiquetado.

Los perfiles específicos del lenguaje y del ambiente de la aplicación definirán casos concretos del estereotipo «dinamic page».

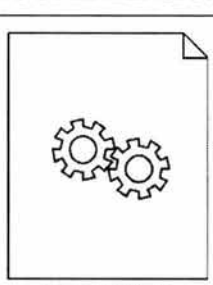

Figura 3-33: Icono «dinamic page».

«Physical Root»

Adicionalmente el paquete «physical Root» contiene los componentes que mapean directamente a los recursos HTTP (es decir páginas estáticas o dinámicas) Estos recursos se pueden solicitar directamente por navegadores del cliente o por otras páginas dinámicas. Es utilizado para resolver referencias URL en la vista lógica, por medio de una relación «link» con una clase que es realizada por un componente residente en el paquete «virtual root».

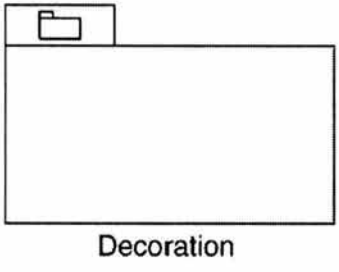

Figura 3-34: «physical Root»

3.2.3 Otros elementos de WAE.

Los estereotipos descritos con anterioridad son suficientes para expresar el diseño de una gran cantidad de aplicaciones WEB,

pero a medida que se agrega complejidad a la aplicación se requerirá de expresar otros elementos que son significativos en la arquitectura, por ejemplo el uso de los marcos en páginas HTML, el manejo de las direcciones de cada página en la aplicación y en otras vinculadas, los servicios WEB, y componentes de etiquetas personalizadas de las páginas de Java Server (JSP), entre otros.

HTML Frames

Los marcos (Frames) permiten al diseñador de Aplicaciones Web dividir una ventana en áreas más pequeñas, cada una de ellas como una página WEB. Un frameset es un tipo especial de página Web que tiene divida su área visible en múltiples ventanas, las cuales contienen su propia página del Web. Un frameset define un arreglo rectangular de marcos en donde cada frameset define el número de filas y columnas en que divide su área visible. Cada frame es una etiqueta potencial. Cada etiqueta es un marco nombrado en un frameset que otras páginas del cliente pueden solicitar a las páginas Web.

Uno de los usos más comunes de los marcos es la implementación de un index frame o tabla de contenido ocupando un marco a la izquierda de la pantalla y mostrando el contenido en el espacio restante, por ejemplo lo que sucede en la interface de Windows Explorer.

«Frameset Class»

Es el estereotipo de la extensión para modela Framesets, abstrae una página HTML que contiene un elemento frameset. Esta

página divide la interface del usuario en marcos o regiones rectangulares que son ejecutadas por páginas cliente distintas. Los frames son opcionalmente identificados por etiquetas.

Debe contener al menos un estereotipo de la clase «Frameset Class» o «target». Sus valores etiquetados Rows y Cols, definen respectivamente el número de renglones y columnas del frame a ser definido.

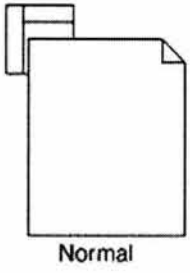

Figura 3-35: «Frameset Class»

«Target Class»

Es la "etiqueta" de un hipervínculo a una «targeted class», de un frame con un nombre especifico en un frameset.

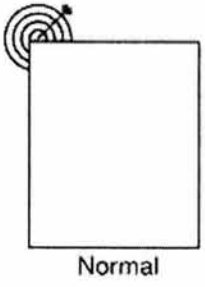

Figura 3-36: «Target Class»

«Targeted link»

Una asociación n-aria, es la etiqueta estándar de una asociación «link» sin embargo, en vez de procesar la solicitud de la página Web en el mismo lugar, esta se envía a la página referenciada en la etiqueta.

Restricciones: Debe hacer referencia exactamente a una clase «target». Debe tener una relación direccional a exactamente una clase Web page: client, server o frameset.

Figura 3-37: «Targeted link»

«Script Libraries».

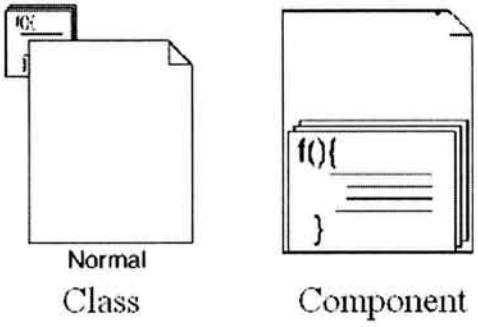

Figura 3-38: Icono para la clase y el componente script library.

Cuando muchos scripts son utilizados en diferentes páginas dentro de un mismo sistema, es mejor crearlos y mantenerlos separados en archivos script library (.js), los cuales contienen funciones del JavaScript que puedan ser incluidos por el cliente,

haciéndose más fácil mantener los script largos y complejos. Los estereotipos de la clase y del componente script library siguientes se utilizan para modelar estos archivos.

Como clase, define un número de funciones y variables JavaScript, y como componente realiza una o más clases «script library» en la vista lógica del sistema a modelar.

«Client script object».

JavaScript no es un lenguaje orientado a objetos puro, pero puede ser tratado como tal, así los objetos JavaScript son modelados con el estereotipo de clase «Client script object»y sus atributos y operaciones son modelados por los atributos y operaciones de la clase.

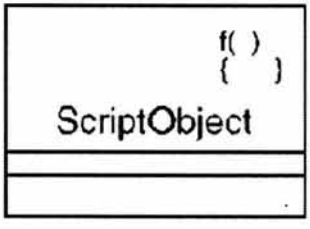

Figura 3-39: «Client script object».

«Virtual root».

Una de las responsabilidades de un servidor WEB es asignar los URL solicitados por los clientes con recursos físicos en el servidor, por ejemplo: un archivo estático, un script dinámico, o un componente compilado, entre otros. En los sistemas más simples existe una relación de un recurso por cada archivo, y cada directorio WEB se asigna a una clase «physical root».

Cuando existen recursos que no son localizados en el mismo sistema de archivos, se agrega una jerarquía virtual a la aplicación, y se representa por el estereotipo «virtual root».

El paquete «virtual root» es usado para resolver referencias en la vista lógica del sistema y es visible al cliente. No tiene restricciones y tiene dos valores etiquetados opcionales: Host y Port.

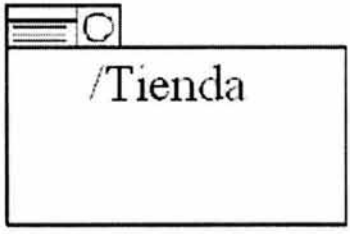

Figura 3-40: «virtual root»

«HTTP resource»

Un componente real que esta asignado a un URL que puede ser requerido por el sistema, no tiene restricciones y un solo valor etiquetado opcional: Filter, que es una expresión regular que es usada para asignar un conjunto de URL's.

Ilustración 3-41: «HTTP resource»

Los recursos HTTP permiten realizar el modelo UX por medio de pantallas, las cuales se alternan entre varios elementos lógicos del modelo del diseño: páginas del servidor y páginas del cliente.

«Screen» class.

Una pantalla es una abstracción de una página WEB en el navegador de un cliente, representa la vista final de la interface del usuario, son parte del modelo UX, en arquitecturas donde la jerarquía de recursos físicos y la virtual URL son diferentes pueden ser utilizadas para realizar pantallas con recursos HTTP.

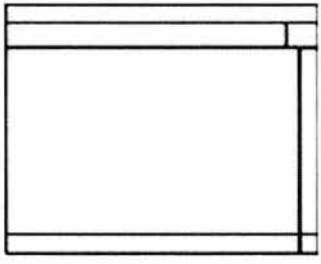

Figura 3-42: Icono «Screen»

«Screen compartment»

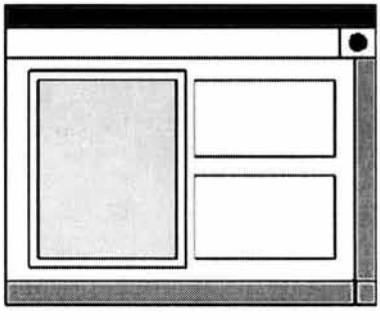

Figura 3-43: Icono «Screen compartment»

Los compartimentos de las pantallas son una parte importante para el modelo UX, ya que al combinarse con otros compartimentos y/o formularios forman páginas simples del sistema, la principal aportación de la clase «Screen

compartment» es el reuso, ya que puede formar parte de diferentes páginas. El estereotipo es definido como una clase que contiene operaciones y atributos igual que el estereotipo «Screen class»; con la diferencia de que «Screen compartment» es siempre contenido en una pantalla principal, la cual forma parte del modelo UX completo.

«Input form»

Al igual que la clase «Screen compartment», la clase «Input form» es una región bien defina en una página y puede ser utilizada en diferentes páginas. También forma parte del modelo UX y puede ser realizada por un componente de página dinámica o estática.

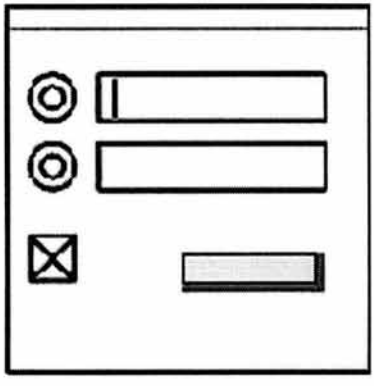

Figura 3-44: Icono «Input form»

Algunos recursos adicionales a las páginas WEB que pueden incluirse en los paquetes de componentes son: «script library», «image» y «stylesheet».

«script library»: se refiere a un archivo que contiene funciones en JavaScript y que son requeridas directamente por el cliente,

«image»: se refiere a un archivo que contiene una imagen que es requerida directamente por el cliente, y «stylesheet» un archivo que contiene definiciones de estilo de la página y que es requerido por el browser.

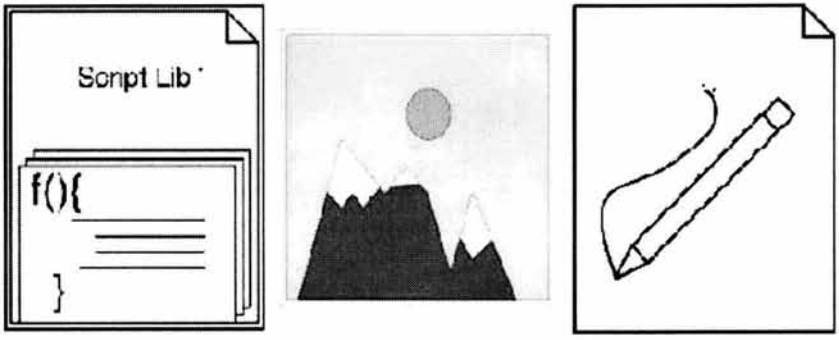

Figura 3-45: «script library», «image» y «stylesheet»

Componentes «JSP»

Una aplicación WEB es una colección de servlets o JSPs que realizan toda la funcionalidad que un usuario solicita. En términos de J2EE[9], una aplicación es definida por el archivo de web.xml, que es un archivo de configuración que contiene las definiciones de la aplicación, incluyendo el servlet mapping, que es cómo el envase del Web en donde se resuelven URLs en servlets o JSPs. Este mapa es una parte dominante de los paquetes componentes «physical root» y «virtual root». Cada componente «JSP»crea un simple archivo JSP (.jsp), localizado en un subdirectorio de la aplicación.

---

[9] J2EE Java 2 Platform, Enterprise Edition.

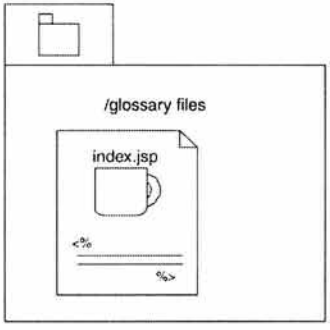

Figura 3-46: Modelado de un JSP dentro de un paquete «physical root»

3.3 Diseño de aplicaciones WEB con la Extensión WAE para UML

Durante la fase del diseño las reglas del negocio se convierten en abstracciones encaminadas a convertirse en software, para el caso de aplicaciones WEB y utilizando la metodología propuesta por Conallen, el diseño inicia con la información obtenida de los modelos del análisis y UX además de los documentos de la arquitectura del sistema; y las actividades de esta etapa giran alrededor de las clases y los diagramas de interacciones.

En aplicaciones WEB, al igual que cualquier sistema cliente/servidor, las actividades del diseño incluyen: repartir los objetos dentro de las capas del sistema y el desarrollo de la infraestructura necesaria y las clases de apoyo para completar el modelo del análisis, pero considerando objetos relevantes las páginas WEB.

Los objetos pueden residir en el servidor, en el cliente o en ambos, repartirlos adecuadamente es una actividad critica y

depende de la arquitectura de la aplicación, En una aplicación WEB compleja, los objetos de validación de entrada de datos, las robustas interfaces especializadas del cliente, por ejemplo pueden ejecutarse en el cliente, pero los objetos que contienen datos, como por ejemplo una lista de clientes, un catalogo de productos deben existir solo en el servidor. Algunos objetos, tales como una factura, pueden tener vidas en ambos.

Las aplicaciones WEB son esencialmente aplicaciones distribuidas, y la distribución de los objetos en la aplicación depende en mucho de la naturaleza de cada uno de los objetos, es importante colocar objetos del lado del cliente, si son llamados más rápidamente, deben colocarse donde su funcionamiento es más efectivo, y podría decirse como regla general que deben existir donde el acceso a los datos sea más fácil y la colaboración entre los objetos permita que las responsabilidades de estos sean ejecutadas.

Posterior a la definición de la arquitectura del sistema deben definirse los elementos requeridos para el modelo UX, describiendo las páginas WEB y sus elementos, así como sus relaciones con otras páginas y otros elementos del sistema. El primer modelo no debe ser muy detallado, no es para la generación de código, nos sirve para identificar nombres, contenidos importantes, campos de entrada, etcétera.

Para completar el modelo UX, las páginas y sus componentes pueden ser expresados en términos de diagramas de clase, y diagramas de interacción que describan los posibles escenarios del sistema.

Diseñar aplicaciones WEB que tienen páginas dinámicas requiere atención al momento de repartir de los objetos, sobre todo en aplicaciones robustas del lado del cliente, donde pueden existir toda clase de objetos y actividades relacionados con los objetos del lado del servidor.

En general el diseño se centra en diagramas de clases y de interacciones, generando un modelo que puede ser traducido directamente en código.

La vista de componentes permite visualizar componentes, interfaces y sus relaciones expresando su distribución en el sistema y en los módulos físicos y ejecutables.

Capitulo 4:
CASO PRÁCTICO TIENDA VIRTUAL "FASHION POL"

En la actualidad, los negocios que tienen éxito son aquellos que procuran estar a la vanguardia tecnológica. La reducción de los costos de los equipos de cómputo, los avances en las servicios de comunicaciones, y el desarrollo de software para Internet, han hecho que la forma de hacer negocios cambie; por esta razón, son cada día más las empresas que tienen presencia en la red, pero Internet, no esta hecho sólo para empresas grandes, de hecho el poseer una empresa pequeña o mediana, o que los artículos que se comercializan no sean de los populares para su venta en la red, no significa que deba ignorarse como un medio de ventas viable para la empresa en cuestión.

Para Fashion Pol, al igual que para muchas empresas, el colocar una tienda virtual, ofrece la posibilidad de incrementar sus clientes potenciales, y con ello las ventas registradas en los locales físicos, el presente trabajo nos ha mostrado una de las etapas dentro de la construcción de aplicaciones WEB, este capitulo muestra el estado existente en Fashion Pol previo a la construcción de la tienda virtual y el modelo que resulta del análisis realizado.

4.1 Planteamiento del problema.

4.1.1 Antecedentes.

Fashion Pol surgió en el estado de Puebla como un taller de maquila en 1983, y se convirtió en una fábrica de ropa de mezclilla para dama en 1986.

Inicialmente se realizaba la distribución de mercancías directamente de la fabrica a tiendas minoristas en los estados de Puebla, Veracruz, Oaxaca, y México principalmente. Se establecieron los primeros dos puntos de venta fijos en 1986:

1. San Martín Texmelucan, Puebla.

2. Chinconcuac, Estado de México en 1986.

Actualmente las instalaciones se encuentran en la delegación Azcapotzalco dentro del Distrito Federal, y existen dos puntos de venta en el centro histórico de Cd. de México:

3. Mercado Abelardo L. Rodríguez (inicio de operaciones en 1993).

4. Plaza Banderas (inicio de operaciones en 1995).

4.1.2 Modelo del Sistema Actual.

La organización actual de Fashion Pol se puede observar en la figura 4.1.

Dentro de la empresa los procesos de producción y ventas son actualmente controlados por procesos manuales.

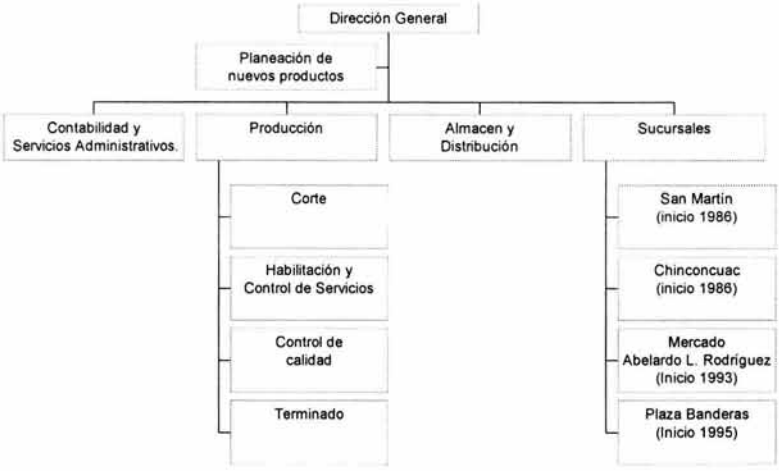

Figura 4-1: Organigrama actual de Fashion Pol.

El almacén de la planta de producción debe tener inventario cero, ya que cuando un modelo se termina e ingresa al almacén, se distribuye en su totalidad a las tiendas para su venta.

4.1.3 Problema.

Las sucursales ubicadas en el centro histórico de la Ciudad de México, han presentado desde hace tres años una reducción importante de los ingresos por ventas. Según los registros contables, los ingresos de diciembre de 1999 a diciembre del 2002, se han reducido hasta en un 45%.

Las principales causas son la inseguridad en el centro histórico de la Ciudad de México y la situación económica del país.

Esto se refleja en la reducción de clientes de mayoreo y el incremento de los clientes de menudeo que acuden con mayor frecuencia al centro en busca de mercancía a precios de mayoreo.

La consecuencia principal es el lento desplazamiento de mercancías, esto a su vez retrasa la recuperación de la inversión. Si esta tendencia no se revierte, podría implicar el cierre de alguna de las sucursales.

Para evitar esto se propone incursionar en el comercio electrónico a través de una tienda virtual. Entre las consecuencias positivas de poner en marcha la tienda virtual están el incremento en ventas, la publicidad y el reconocimiento para la marca.

4.2 Análisis.

4.2.1 Definición de los límites del problema.

La presente propuesta, implica automatizar los procesos de venta de las tiendas e incluir una tienda virtual sumándola a los puntos de venta ya existentes, esto implica la creación de un sistema muy complejo cuya aplicación deberá incluir virtualmente todas las actividades que intervienen en el proceso de venta de mercancías de la empresa, parte del reto que conlleva el desarrollo de un sistema tan amplio es que la planificación se realice considerando todos los procesos de la empresa.

Para delimitar el ámbito de la tienda virtual nos limitaremos a considerar como plan operacional de la tienda virtual, exclusivamente las siguientes actividades funcionales del negocio:

- Atención al cliente: Responsable de recibir pedidos por parte del cliente y responder a

preguntas de este sobre el estado de un pedido.

- Contabilidad: Responsable de enviar facturas y controlar los pagos de los clientes.
- Almacén: Responsable de recibir mercancías del almacén central de la fabrica, de ensamblar los paquetes para envió de mercancías a los clientes, y de la ubicación de las mercancías dentro del almacén.

Para el caso del diseño de la tienda virtual Fashion Pol con la extensión WAE de UML, se pondrá especial atención a las capas de presentación de la interfaz del usuario, presentando el modelo dentro de esta frontera y solo en algunos casos incluyendo los elementos de todas las capas.

4.2.2 Arquitectura de la aplicación.

Para el desarrollo de esta aplicación se consideran los siguientes elementos como significativos para la arquitectura:

- Clases JSP y servlet.
- Código del lado del cliente.
- JavaBean[1] usados en JSPs.

---

[1] JavaBeans son componentes que se conectan o desconectan de las páginas WEB con relativa facilidad.

La tienda virtual Fashion Pol permitirá al usuario navegar en un sitio de Internet, y revisar un catalogo de prendas de vestir para dama y seleccionar alguno(s) para su compra, y para el caso de pagos en línea deberá garantizarse la integridad de los datos proporcionados por el cliente por medio de un sistema de pago seguro basado en SSL.

La tienda virtual Fashion Pol se estructura como una aplicación de tres capas con la interfaz del usuario desarrollada en HTML en la primera, la lógica de la aplicación en la segunda será desarrollada en el lenguaje Java, y la tercera contiene los datos almacenados en una base de datos a la que se accede mediante JDBC.

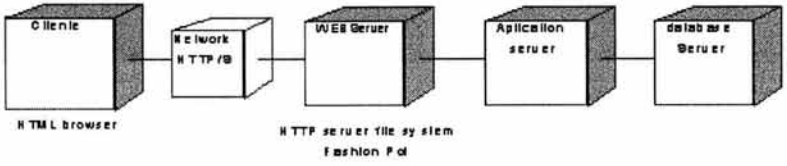

Figura 4-2: Arquitectura de tres .

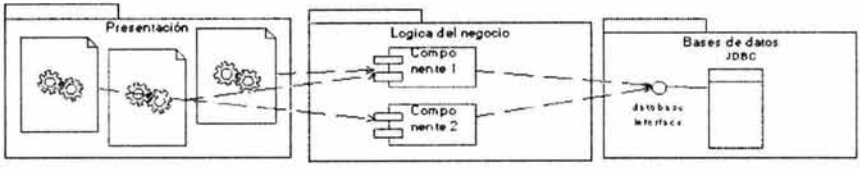

Figura 4-3: Distribución de los objetos por capas.

4.2.3 Requisitos de la Tienda Virtual Fashion Pol.

Los requisitos que debe cubrir obligatoriamente la tienda virtual son:

1. Conectar a un sistema de ventas en las tiendas, para tener un inventario en tiempo real dentro de la tienda virtual.
2. Tener un módulo de administración de pedidos.

3. Mostrar productos con foto, precio y características.
4. Proporcionar seguridad en las transacciones.
5. Permitir el uso de diferentes opciones de pago.
6. Tener carro de compras.
7. Ser de rápido acceso.
8. Ser de fácil mantenimiento.
9. Tener espacio suficiente en disco para catálogo de productos y clientes.
10. Contar con la autorización de uso de todo el software involucrado.

Los requisitos deseados dentro de la tienda virtual son:

1. Utilizar gráficos de alta calidad.
2. Dar seguimiento a los pedidos por parte del cliente y la administración.
3. Mantener un histórico de las transacciones por cliente para facilitar las labores de atención al cliente.

4.2.4 Requerimientos funcionales, no funcionales y de dominio.

Requerimientos funcionales.

1. El sistema permitirá dar mantenimiento al catálogo de productos.
2. El sistema mantendrá un catálogo de clientes.
3. El sistema aceptará pedidos en Internet.
4. El sistema permitirá al cliente seleccionar artículos del catálogo de productos.
5. El sistema proporcionará protección de contraseña para todos los clientes.
6. El sistema permitirá al cliente seleccionar la forma de pago.

7. El sistema proporcionará medios seguros para permitir que los clientes paguen con tarjeta de crédito.
8. El sistema proporcionará un número de cuenta que este preautorizada para pagar vía depósito bancario.
9. El sistema proporcionará medios seguros para permitir que los clientes paguen vía depósito bancario.
10. El sistema permitirá al cliente corregir información antes de finalizar una transacción.
11. El sistema proporcionará un enlace entre el pedido y el estado del envío.
12. El sistema proporcionará una interfaz entre el sistema de pedidos y el inventario de la tienda.
13. El sistema proporcionará reportes de las transacciones realizadas.

Requerimientos no funcionales.

1. Permitirá integrar los servicios bancarios necesarios para la realización de las transacciones.

2. Cumplirá la normatividad de comercio electrónico vigente.

- La información proporcionada por el cliente será tratada con confidencialidad, no podrá ser difundida o transmitida a otros proveedores externos, salvo autorización expresa del cliente, o por requerimiento de autoridad competente.

- Se informará al cliente de las características generales de los elementos técnicos utilizados para brindar seguridad y confidencialidad a la información que proporcionará.

- Se notificará al cliente antes de finalizar una transacción del domicilio físico, teléfonos, y demás medios donde pueda acudir para presentar quejas y/o reclamaciones.

- Se notificará al cliente de los términos, condiciones, costos y en caso necesario, de los cargos adicionales, formas de pago de los productos y servicios ofrecidos por Fashion Pol.

- La información proporcionada al cliente, acerca de los productos, será clara y suficiente para evitar que el cliente pueda resultar engañado.

Requerimientos de dominio.

- El cliente podrá seleccionar si desea imprimir nota de remisión o factura al realizar una compra, y en caso de seleccionar factura, está será impresa de acuerdo a la legislación vigente.

4.2.5 Requerimientos del usuario.

Para actualizar el inventario:

Cuando un nuevo modelo es iniciado en el proceso de producción debe actualizarse el catálogo de productos, indicando la clave del producto, el nombre, una descripción del mismo; así como determinar si el modelo será controlado por talla y/o color.

Cuando se registre una entrada del almacén a una tienda, se entregará al encargado de la tienda una nota de entrada de almacén, una vez que se verifique la entrada con el contenido de la nota, el administrador de la tienda entrará al módulo de inventario y proporcionará su nombre de usuario y clave para ser

identificado y podrá alimentar la nota de entrada, tecleando el número de la nota, la fecha de entrada, la clave del producto, la cantidad que se recibe por talla y por color.

Dentro de la tienda física:

El cliente, seleccionará las mercancías a comprar y el encargado de la tienda o el vendedor entrara al sistema de ventas, se identificará con un nombre de usuario y contraseña y podrá registrar la compra, introduciendo la clave de la mercancía, la cantidad, talla y color; el sistema asignará el precio de acuerdo al número de piezas, si es una o dos, asignará precio de menudeo, y si son más de tres, precio de mayoreo; también se asignará un consecutivo a la remisión que se imprimirá. Una vez terminada la captura de la compra se actualiza la cantidad de piezas disponibles por modelo, talla y color en el almacén. En las tiendas no se aceptan devoluciones de mercancía, solo se hacen cambios, y los cambios se hacen solo durante los quince días posteriores a la venta, por lo tanto estos serán aceptados al realizar una compra, introduciendo él numero de la remisión utilizada en la compra anterior y se podrán seleccionar los artículos a devolver, solo de los artículos de dicha remisión, siempre y cuando la remisión no sea anterior a quince días. En caso de que una mercancía ingrese a la tienda después de un cambio, se deberá actualizar la existencia de dicho producto.

Dentro de la tienda virtual:

Cuando un cliente ingresa a la tienda virtual y desea hacer una transacción por primera vez, se le asignará un nombre de usuario y una clave que podrá utilizar en las siguientes transacciones, para identificarse, existirá una pasarela con modelos en movimiento,

cuando uno sea seleccionado, se mostrarán detalles del modelo y vistas de la prenda, si alguno es del agrado del cliente y existe la disponibilidad en talla y color deseado podrá ser adicionado al carro de compras, se podrá iniciar un pedido, seleccionando la forma de pago, de entre las opciones de pago con tarjeta crédito en línea, o pago en banco a una cuenta, si se selecciona esta opción se le proporcionará una cuenta bancaria y una clave de transacción al cliente para que realice el deposito en banco, se llenará la forma con los datos de la transacción, dirección para realizar la entrega de la(s) mercancía(s), cuando los datos se hayan terminado de introducir, el cliente podrá verificarlos, y en caso de desear hacer una corrección se regresará a la forma de captura de datos, una vez finalizada la transacción se procederá en el sistema de la tienda a marcar la mercancía como vendida, para evitar que la misma sea vendida nuevamente, si el pago se realiza a través de un deposito bancario, cuando este sea realizado, el cliente puede informar por correo electrónico los datos del deposito, o bien serán proporcionados por el banco a través de algún sistema de recepción automática de pagos que identifique cuando la clave de la transacción sea realizada.

Reportes de transacciones:

Cuando el administrador de una tienda, decida hacer una consulta de las transacciones realizadas por Internet, entrará al sistema identificándose con un nombre y una, escogerá el módulo de reportes, y seleccionara el reporte deseado de entre los siguientes: Transacciones realizadas por un rango de días, los pedidos realizados y su estado (en espera de pago, en transito de mercancía, o entregado), ventas realizadas por modelo, talla y/o color.

Cuando el administrador de una tienda, decida hacer una consulta de las transacciones realizadas en el sistema de ventas

de la tienda, entrará al sistema y se identificará con un nombre y una clave para entrar al sistema, y escogerá el módulo de reportes, y seleccionara el reporte deseado de entre los siguientes: Transacciones realizadas por un rango de días, consecutivo de remisiones por día, ventas realizadas por modelo, talla y/o color.

4.2.6 Requerimientos del sistema.

Proceso:	Alta de producto.
Descripción:	Asignar códigos a los modelos nuevos.
Entrada:	Descripción del modelo, características, colores, rango de tallas.
Fuente:	Orden de producción.
Destino:	Actualizar catálogo de artículos.
Salida:	Código del artículo.
Requerimientos:	El nuevo modelo debe contar con orden de producción.
Proceso:	Entrada al almacén.
Descripción:	Actualizar los artículos disponibles para venta de acuerdo a las entradas del almacén central.
Entrada:	Código del producto, colores, rango de tallas, número de piezas que ingresan.
Fuente:	Entrada de Almacén.
Destino:	Actualizar catálogo de artículos.
Salida:	Total de piezas disponibles para venta.
Requerimientos:	El artículo que ingresa debe estar dado de alta.

Proceso:	Alta de cliente.
Descripción:	Actualizar el catálogo de clientes.
Entrada:	Datos del cliente.
Fuente:	Cliente.
Destino:	Actualizar catálogo de clientes.
Salida:	Nombre de usuario del cliente y password.
Requerimientos:	El usuario debe entrar a la página.
Proceso:	Solicitud de pedido.
Descripción:	El cliente solicita una mercancía para que le sea enviada, confirma su solicitud y realiza pago o reserva el pedido hasta para hacer depósito.
Entrada:	Nombre y password del usuario, Código del producto, colore(s), talla(s), número de piezas que solicita.
Fuente:	Cliente, catalogo de productos.
Destino:	Actualizar existencia de artículos y pedidos.
Salida:	Actualización de inventario, contabilidad.
Requerimientos:	El cliente debe estar dado de alta, el producto debe contar con existencias para venta.
Proceso:	Selección de forma de pago.
Descripción:	El cliente ha realizado un pedido, el sistema ha calculado un importe a pagar y el cliente selecciona una forma de pago.
Entrada:	Nombre y password de usuario, tipo de pago.
Fuente:	Cliente.
Destino:	Actualizar estado del pedido.
Salida:	Estado del pedido.
Requerimientos:	El cliente debe estar dado de alta, debe existir un pedido a pagar.

Proceso:	Selección de reportes.
Descripción:	El administrador de la tienda desea hacer consultas al sistema..
Entrada:	Nombre y password del administrador, número de reporte seleccionado, Rango de datos a solicitar.
Fuente:	Administrador.
Destino:	Reporte por impresora o pantalla.
Salida:	El administrador debe tener privilegios dentro del sistema para la elaboración de reportes.
Requerimientos:	

4.2.7 Requisitos de seguridad.

Aunque se han mencionado algunas características de seguridad de la tienda es de destacarse que debe controlarse el modo de acceso a la tienda, el cual puede realizarse como cliente y como administrador (o usuario) de la tienda, en el primer caso el cliente podrá acceder a la tienda de forma anónima hasta el momento de realizar una compra, en cuyo caso se le solicitará se identifique con un nombre de usuario y una contraseña, a efectos de que no sean solicitados sus datos cada vez que ingrese a la tienda y faciliten el rastreo de los pedidos; como administrador (o usuario) se solicitara un nombre de usuario y una contraseña, la cual estará vinculada a los privilegios del usuario para las operaciones a realizar, el control de los usuarios y sus privilegios estará a cargo del administrador de la tienda.

Dado que Internet es una red pública y es factible que con cierta facilidad se intercepten datos por tercero, es importante realizar un traslado y resguardo seguro de la información personal del

cliente, en particular la relacionada con su tarjeta de crédito, por lo tanto esta está no deberá permanecer almacenada en la base de datos, y ser solicitada en una compra y ser utilizada para esa compra en particular.

4.2.8 Clases del análisis.

Una primera versión de las clases que deben componer la aplicación de la tienda virtual Fashion Pol, se observa en el siguiente diagrama de clases.

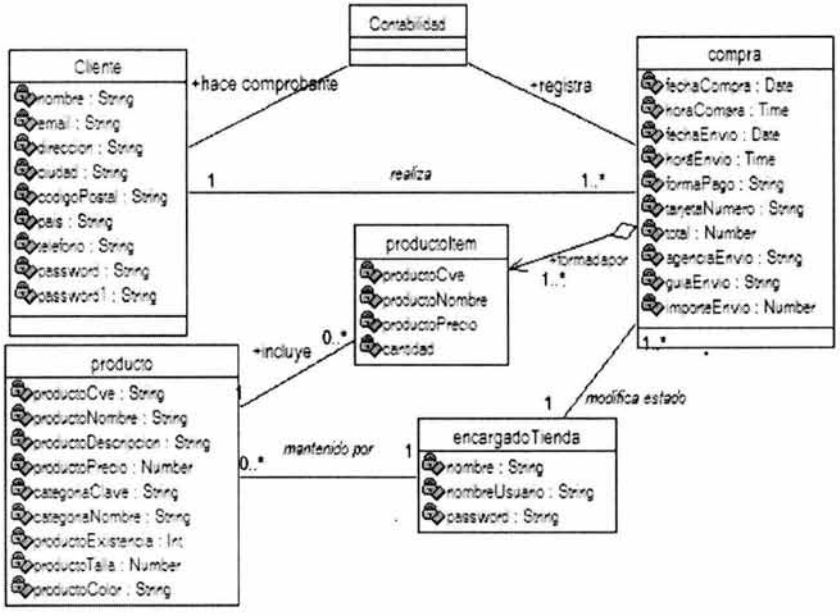

Figura 4-4: Clases del análisis.

4.3 Diseño.

El modelo de diseño es un modelo que se obtiene de la evolución de otros modelos en uno más completo, para el caso de emplear la extensión WAE al diseño de aplicaciones se relaciona con la

vista de casos de uso y el modelo UX, haciendo énfasis en la capa de la presentación y la interfaz del usuario.

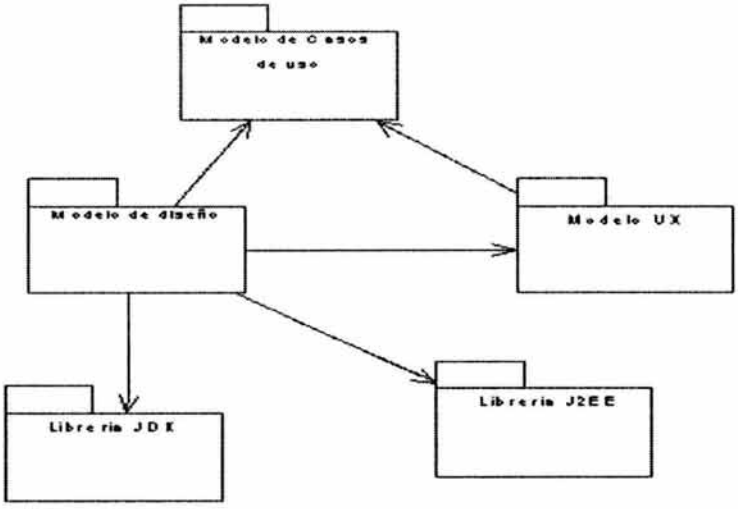

Figura 4-5: Resumen de modelos y paquetes de la aplicación.

4.3.1 Vista de casos de uso.

Para la aplicación de la Tienda Virtual Fashion Pol se distinguen cinco actores: cliente, administrador de la tienda, contabilidad, servicio de entrega y centro autorizador.

El cliente es el principal actor de la aplicación y representa a cualquier usuario de Internet quien puede navegar por la aplicación y realizar una compra.

El administrador del sistema, será la persona encargada de dar mantenimiento a los catálogos y obtener reportes del funcionamiento de la aplicación.

Contabilidad, el encargado de registrar las compras en la contabilidad, y emitir los comprobantes de la compra a los clientes.

Servicios de entrega, quien estará a cargo de realizar los envíos de mercancía a los clientes.

Centro autorizador, quien en caso de requerirse una autorización para una transacción de pago con tarjeta de crédito la emitirá con fundamento en la información proporcionada por el cliente.

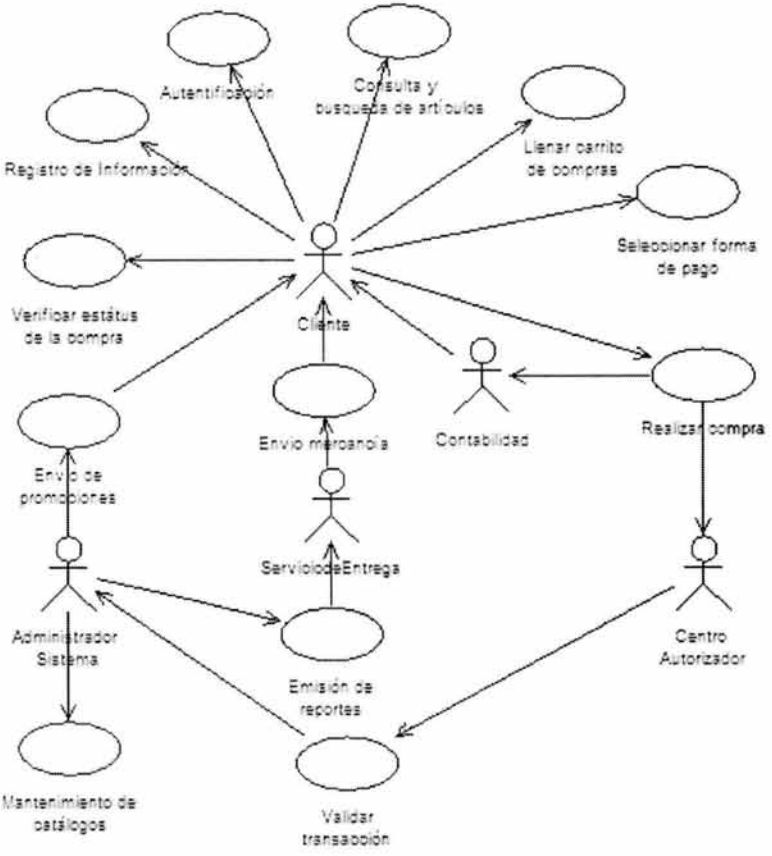

Figura 4-6: Diagrama de casos de uso.

4.3.2 Modelo UX.

El modelo UX presenta información de la aplicación a desarrollar desde el punto de vista de las pantallas de información presentadas al usuario, las dos principales clases que se muestran son «screen» y «form».

Durante este modelo se usa el signo de pesos ($) para indicar que la pantalla es accesible desde cualquier otra pantalla del sistema, y el signo de más (+) indica que en esa pantalla se muestra una lista de ítems, uno a la vez. Estas convenciones se usan en este modelo en particular y no forman parte de UML ni de la extensión WAE.

Figura 4-7: Flujo básico Tienda Virtual Fashion Pol

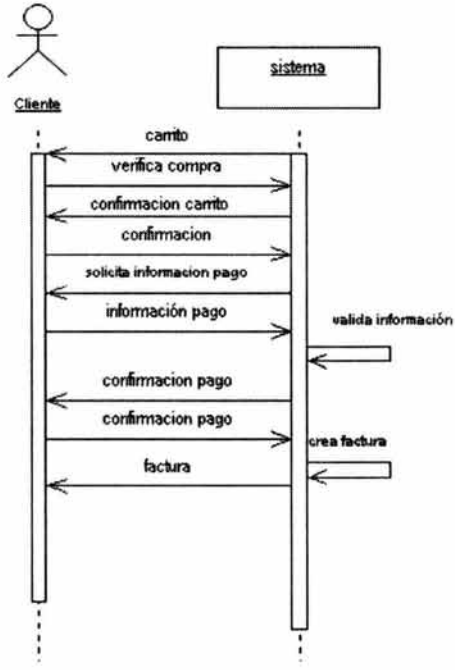

Figura 4-8:Flujo básico Tienda Virtual Fashion Pol (Extensión compra producto)

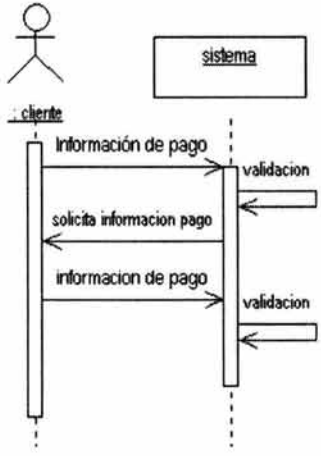

Figura 4-9: Flujo alternativo: Validación de información de pago.

La figura 4-7 muestra una ruta de navegación básica para un usuario de Internet, quien puede solicitar la pagina principal de la tienda virtual y navegar por las diferentes páginas que la componen, este diagrama se extiende al que se muestra en la figura 4-8 para el cual a su vez, existe un flujo alternativo para la validación de la información relacionada con los pagos con tarjeta.

La figura 4-10 muestra el mapa de navegación de primer nivel para la aplicación la tienda virtual Fashion Pol.

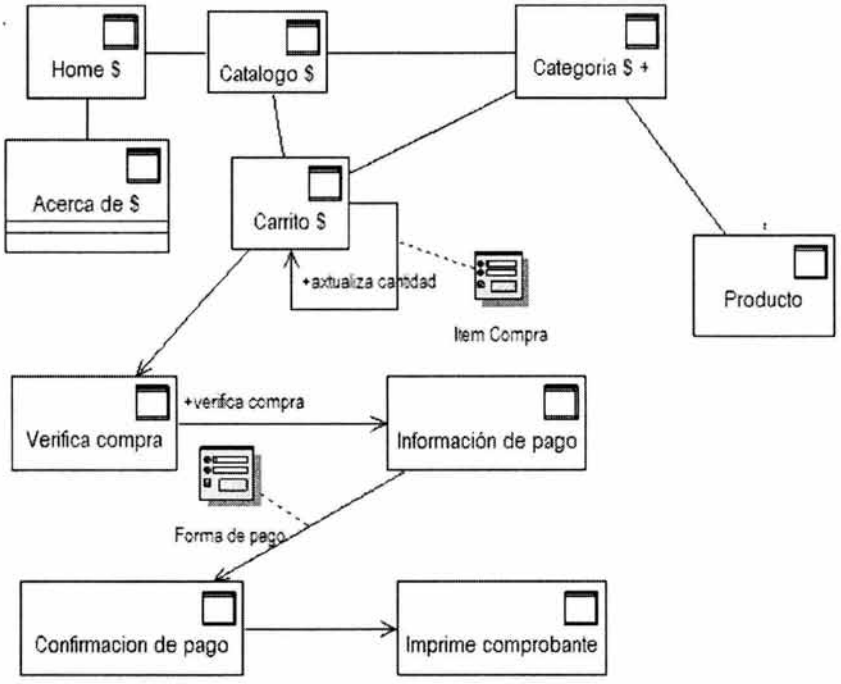

Figura 4-10: Mapa de navegación de primer nivel.

De la figura 4-10 se extrae el detalle de navegación que se muestra en la figura 4-11.

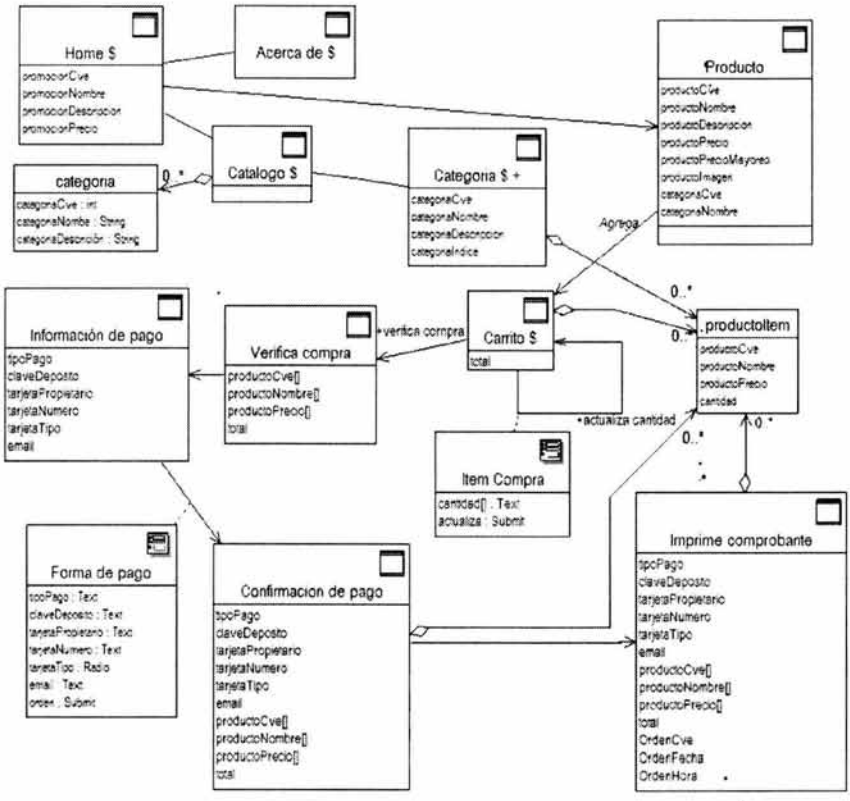

Figura 4-11: Detalle de navegación.

Los siguientes diagramas de secuencia muestran el flujo de las pantallas en el sistema para el caso de que un cliente seleccione una oferta de la página principal y decida comprarlo (figura 4-12), y para el caso de que se decida consultar el catalogo de productos, seleccionar alguno y comprarlo (figura 4-13).

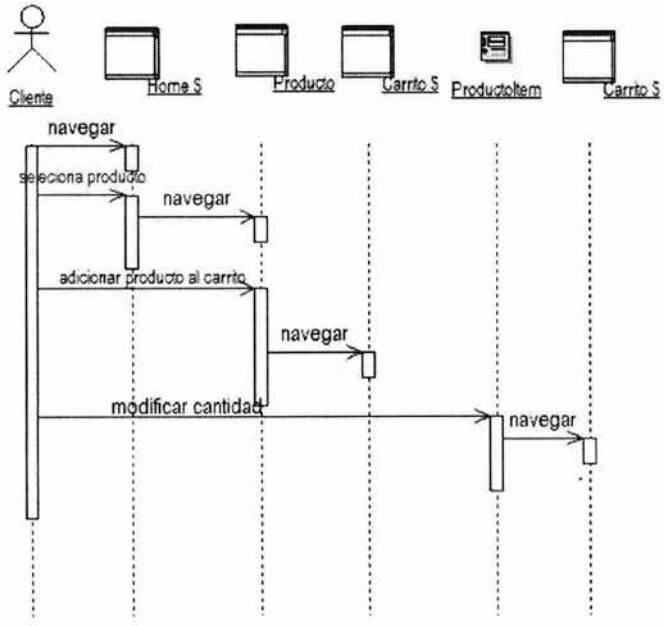

Figura 4-12: Escenario compra de oferta.

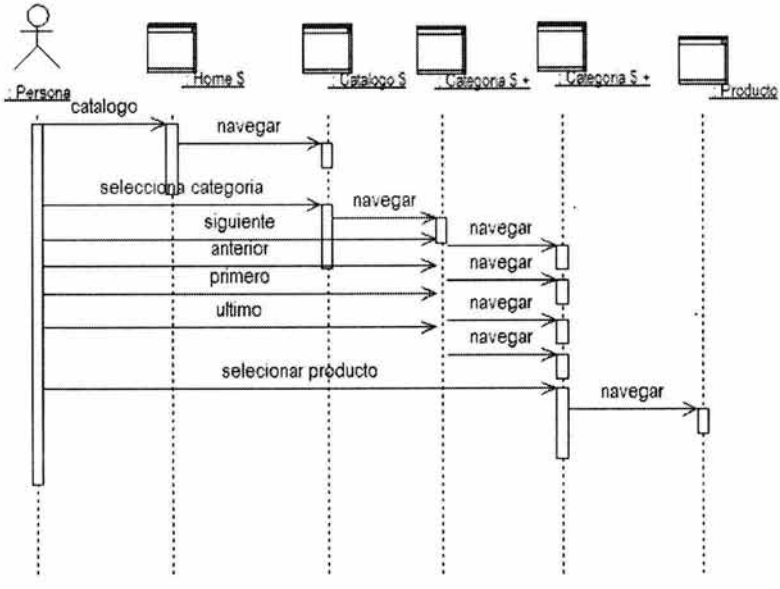

Figura 4-13: Escenario: Compra producto de catalogo.

4.3.3 Módulos de la aplicación.

La aplicación a desarrollar se dividirá en tres módulos, dependiendo del tipo de usuario que accede al mismo.

1. **Módulo de clientes**: este permitirá realizar búsquedas y compras en la tienda, y será el módulo responsable de mantener informado a los clientes del estado de sus compras.

2. **Módulo de encargado de almacén**: En este módulo se permitirá al encargado de la tienda dar mantenimiento al stock de los productos y administrar y consultar las compras que los clientes van realizando.

3. **Módulo de punto de venta**: Este módulo será el encargado de permitir que los vendedores que se encuentran en la tienda alimenten las ventas que hacen en la misma, al momento de realizarlas, esto permitirá que el inventario que existe en Internet se mantenga actualizado.

4.3.4 Diagramas de clases para los módulos de la aplicación.

Clases nuevas.

Se crea la clase **Persona** y de ella se heredan los clientes y responsable de la tienda, de almacén, y de envíos. Para que la aplicación soporte la presentación de precios en varios tipos de moneda se crea la clase **Moneda** que almacena la información relacionada a las monedas y la clase equivalente **MonedaUsuario**, que almacena la información de la moneda utilizada por cada usuario en un momento determinado, como

moneda principal se utilizara el peso mexicano y los factores de cambio de otras monedas serán con relación a esta última.

Módulo de clientes.

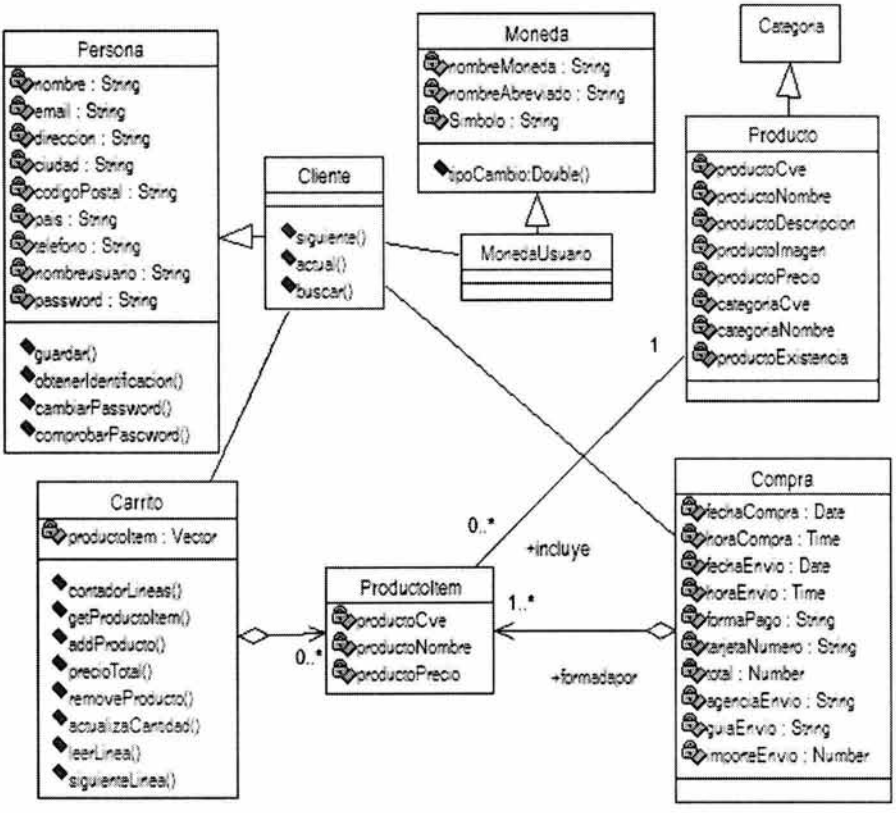

Figura 4-14: Modulo de clientes.

Este módulo permitirá a los clientes que navegan por Internet:

- Ingresar a la tienda, buscar y navegar productos dentro de las clases que forman la tienda virtual, teniendo la posibilidad de añadirlos al carrito de compra.

- Realizar consultas sobre las compras que han realizado anteriormente, para conocer su estado, los productos solicitados, la fecha de recepción, envío, etcétera.

- Darse de alta como cliente del centro comercial o, si ya esta dado de alta modificar su información personal almacenada en el sistema.

- Realizar la compra de los productos que tenga actualmente en su carrito, proporcionando los datos relativos al envío y cobro de la compra.

Módulo de encargado de almacén.

Este módulo será el encargado de:

- Dar mantenimiento al catalogo de artículos dando de alta productos nuevos y capturando entradas del almacén central.

- Administrar los pedidos de los clientes, y notificar de los productos que deben ser enviados a los clientes.

- Seleccionar los reportes que el sistema generé de la operación del mismo.

Módulo de punto de venta.

Este módulo será el encargado de:

- Permitir a los vendedores que se encuentran en la tienda física capturar las ventas que hacen al momento de realizarlas.

- El sistema disminuirá el inventario disponible cuando se requiera.

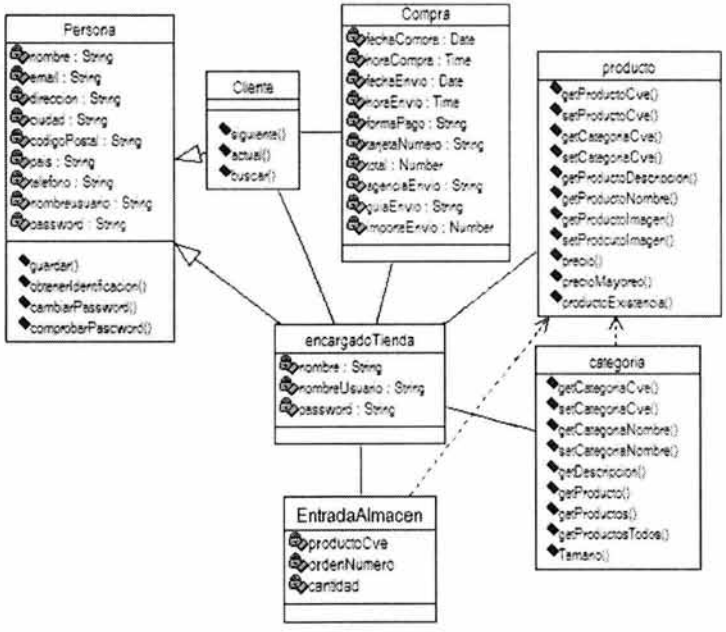

Figura 4-15 Módulo de encargado de almacén

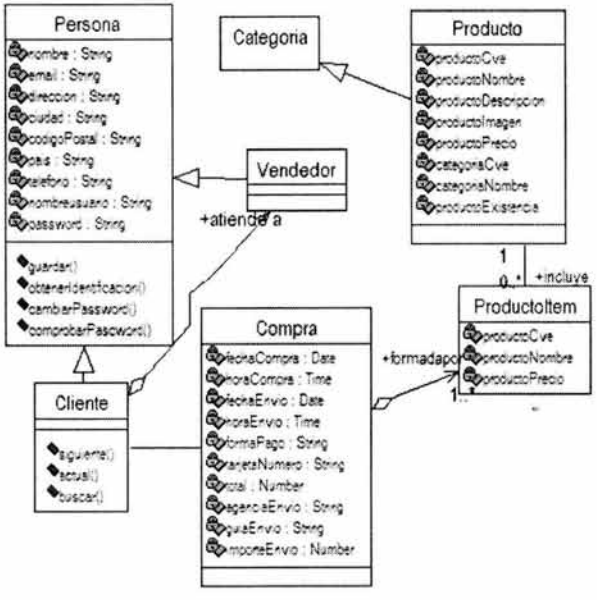

Figura 4-16: Módulo de punto de venta.

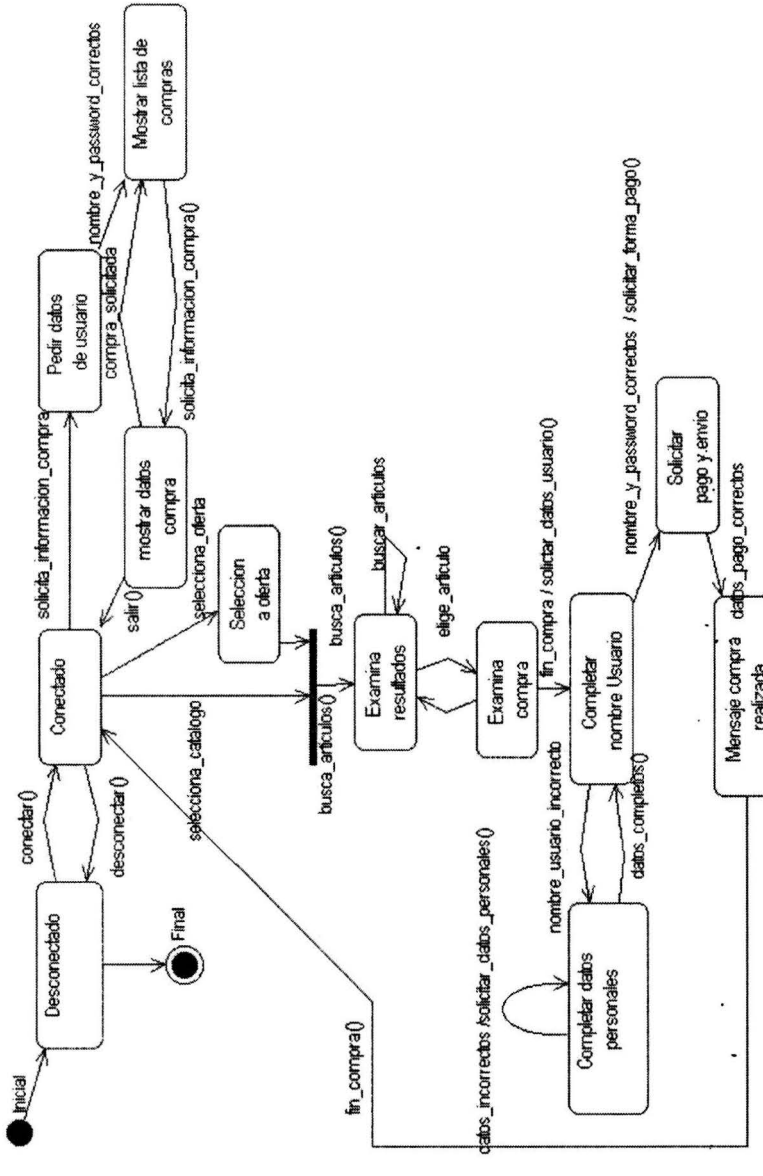

Figura 4-17: Diagrama de estados para la clase cliente.

4.3.5 Capa de Datos.

Esta parte del diseño se centra en hacer corresponder el modelo de clases a la base de datos a utilizar en el sistema.

Clase Persona.

Descripción	De esta clase se heredan los clientes y el personal que tiene acceso al sistema por parte de la tienda, compartiendo la información que se almacenará en ambos casos y los métodos específicos par la administración de las contraseñas de clientes y empleados.
superclase	
Atributos	Privado: nombre: String Privado: email: String Privado: direccion: String Privado: ciudad: String Privado: codigoPostal: String Privado: pais: String Privado: telefono: String Privado: nombreUsuario: String Privado: password: String
Operaciones	Protegida: guardar() Protegida: obtenerIdentificación() Protegida: cambiarPassword() Protegida: comprobarPassword()

Enlaces	

Clase Cliente.

Descripción	Representa a los clientes que mediante la red acceden a la tienda para realizar sus compras o asisten a la tienda física a realizarlas.
superclase	Persona.
Atributos	
Operaciones	
Enlaces	Realiza: se refiere al hecho de que el cliente realiza compras en la tienda.

Clase Empleado.

Descripción	Representa a cada uno de los empleados de las tiendas.
Superclase	Persona.
Atributos	Privado: tipoEmpleado: String (ADMINISTRADOR/VENDEDOR)
Operaciones	
Enlaces	MantenidoPor: Indica que el catálogo de productos es mantenido por el Administrador.

	ObtenidoPor: Indica que el Administrador selecciona los reportes del sistema. RealizaCompra: Captura el hecho de que un vendedor en la tienda física realiza ventas y en el momento de realizarlas captura el movimiento al sistema.

Clase Compra.

Descripción	Representa las compras que los clientes van realizando en el centro comercial.
Superclase	
Atributos	Privado: fechaCompra: Date Privado: horaCompra: Time Privado: fechaEnvio: Date Privado: horaEnvio: Time Privado: formaPago: String Privado: numeroTarjeta: String Privado: agenciaEnvio: String Privado: importeEnvio: Integer Privado: guiaEnvio: String
Operaciones	Publica: guardar (c:conexcion) Protegida: obtenerIdentificacion (c:conexcion)
Enlaces	Realiza: se refiere al hecho de que el cliente realiza compras en la tienda.

	FormadaPor: Agregación que indica que una compra esta compuesta por un conjunto de líneas que la detallan (ProductoItem).

Clase Producto.

Descripción	Representa los productos que se venden tanto en la tienda física, como en la tienda virtual..
superclase	
Atributos	Privado: productoCve: String Privado: productoNombre: String Privado: productoDescricpción: String Privado: productoImagen: Image Privado: productoPrecio: Number Privado: productoPrecioMayoreo: Number Privado: categoriaCve: String Privado: categoriaNombre: String
Operaciones	Publica: getProductoCve () Publica: setProductoCve () Publica: getProductoNombre () Publica: setProductoNombre () Publica: getCategoriaCve () Publica: setCategoriaCve () Publica: getProductoDescripcion () Publica: setProductoDescripcion () Publica: getProductoImagen ()

	Publica: setProductoImagen () Publica: getProductoPrecio () Publica: getProductoPrecioMayoreo ()
Enlaces	Incluye: Captura el hecho de que cada línea (ProductoItem) de una compra o del carrito esta asociada a la petición de un producto.

Clase Carrito.

Descripción	Esta clase sirve para representar que el cliente, cuando visita la tienda virtual, para hacer una compra, va eligiendo los artículos que quiere comprar y los va almacenando en su carrito de la compra de forma temporal, si se concreta la compra los productos del carrito pasan a ser los productos de la compra y el carrito se destruye.
Superclase	
Atributos	Privado: respuestaHTTP: HttpServletResponse Privado: peticionHTTP: HttpServletRequest Privado: valorCookie: String Privado: cookieCarrito: Cookie Privado: lineaCarritoActual: Integer
Operaciones	Publica: añadirLinea (1:ProductoItem)

	Publica: borrarLinea (1:ProductoItem) Publica: leerLineaCarrito(): ProductoItem Publica: borrarContenido() Publica: leerLineas()
Enlaces	Asociasión con el cliente al que pertenece. Agregación para recoger las líneas que lo forman

Clase Categoría.

Descripción	Representa las categorías de los productos que pueden aparecer en la tienda.
Superclase	
Atributos	Privado: categoriaCve: String Privado: categoriaNombre: String Privado: categoriaDescripcion: String
Operaciones	Publica: guardar (c:conexcion) Publica: setCategoriaCve() Publica: getCategoriaCve() Publica: setCategoriaNombre() Publica: getCategoriaNombre() Publica: setCategoriaDescripcion() Publica: getCategoriaDescripcion() Publica: getProducto() Publica: getProductos()

	Publica: getProductosTodos() Protegida: obtenerIdentificacion (c:conexcion)
Enlaces	Contiene: Se refiere al hecho de que una categoría contiene varios productos.

Clase ProductoItem.

Descripción	Sirve para representar a cada una de las líneas que forman parte del carrito de un cliente o de una compra concretada.
superclase	
Atributos	Privado: prodctoCve: String Privado: Precio:Number
Operaciones	
Enlaces	Agregación que indica que un carrito esta formado por varias líneas (ProductoItem) Agregación que indica que una compra esta formada por varias líneas (ProductoItem) Asociación que indica que contiene un Producto.

Clase Moneda.

Descripción	Clase que implementa una moneda, se usa para la conversión de precios a monedas distintas al peso.
Superclase	
Atributos	Publico: nombreMoneda: String Publico: nombreAbreviado: String Publico: simbolo: String Publico: tipoCambio: Double
Operaciones	Publica: siguiente () Publica: actual () Publica: buscar ()
Enlaces	

Clase MonedaUsuario.

Descripción	Clase que permite que el sistema almacene la moneda en que un determinado usuario desea ver los precios de los productos.
Superclase	
Atributos	Publico: nombreMoneda: String Publico: nombreAbreviado: String Publico: simbolo: String Publico: tipoCambio: Double

Operaciones	Publica: leerMoneda () Publica: cambiarMoneda (long)
Enlaces	Asociación con el Cliente para asociar al mismo la moneda que tiene seleccionada. Asociación a la clase moneda, para acceder a las instancias de esta.

4.3.6 Arquitectura modular.

La figura 4-19 muestra la división de la aplicación en los módulos de la misma. Además muestra la relación entre ellos.

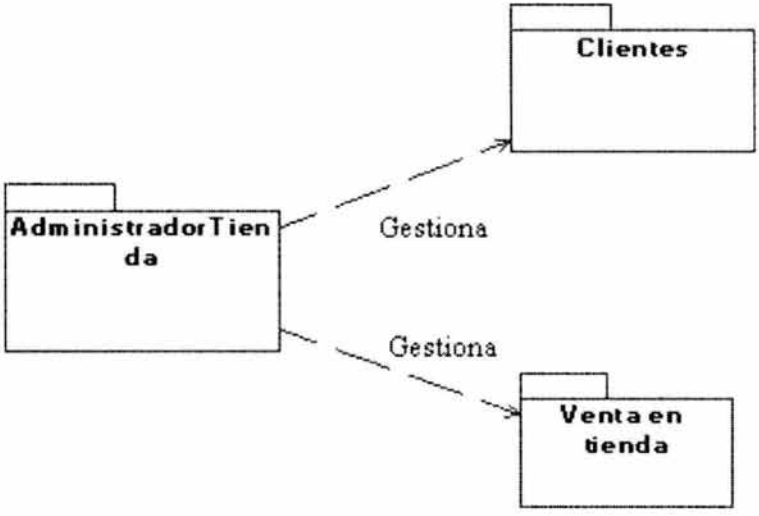

Figura 4-18: Diagrama modular de la aplicación

La siguiente matriz de trazabilidad muestra la relación entre los módulos y las diferentes clases que lo forman. La "X" indica que la clase forma parte del módulo.

Módulos			
Clases	Clientes	Administrador de la Tienda	Ventas en la tienda Física
Persona.	X	X	X
Cliente	X	X	X
Empleado		X	X
Compra.	X	X	X
Producto	X	X	X
Carrito	X		
Categoría	X	X	X
ProductoItem	X	X	X
Moneda	X		
MonedaUsuario	X		

Tabla 4-1: Matriz de trazabilidad de la Aplicación Fashion Pol.

4.3.7 Capa de presentación.

La mayor parte de las aportaciones de la extensión WAE es en la capa de la presentación, esta capa es responsable de responder a los requerimientos de los clientes proveyéndolos del código HTML adecuado para satisfacer a sus peticiones. Los principales componentes de esta capa es el servidor WEB, y los motores JSP.

Cada página en esta aplicación es requerida por un nombre de pantalla, que ha sido definido en el modelo UX, la figura 4-19 muestra la asignación de JSP a cada compartimiento que contribuye a la construcción de la página HOME.

El sufijo .jsp es sólo una convención en este modelo, se usa para distinguir entre páginas clientes y servidores, las cuales frecuentemente tienen nombres similares. Cada página en la aplicación es asignada a un JSP o una página HTML, los cuales son usados para construirlas.

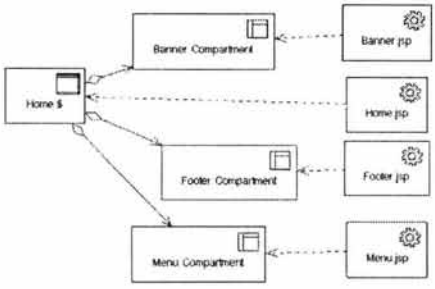

Figura 4-19: Mapeo de compartimentos comunes y archivos JSPs.

Una parte fundamental de esta arquitectura del sistema es el mecanismo que se utiliza para aceptar y procesar los requerimientos entrantes, este mecanismo se deriva de dos patrones arquitectónicos de aplicaciones WEB, Controlled Controllers[2] y Master Template[3].

El manejo de los requerimientos de páginas en esta aplicación empieza por un mapeo del archivo web.xml que redirecciona todos los requerimientos entrantes de páginas con la extensión .src a una instancia de la clase RequestProcessor. El request processor es un servlet y dentro de este modelo es clase estereotipada <<server page>>.

---

[2] El mecanismo controlled controllers permite diseñar una aplicación alrededor de múltiples controladores discretos en la capa de presentación. Cada controlador es un objeto simple, el cual puede tener objetos auxiliares, pero no necesarios, todos ellos soportados por una interface común, El elemento clave de este mecanismo es que el servidor de aplicaciones WEB, es configurado para dirigir todos los requerimientos a través de una simple clase, una instancia para cada sesión iniciada por un usuario, este objeto es el RequestProcessor.

[3] Master Template Pattern es un mecanismo que utiliza una pagina como plantilla JSP para todas las páginas que salen, de esta forma se logra una interface del usuario consistente y proporciona un solo código a actualizar.

La figura 4-20 muestra las principales clases y JSPs involucrados en procesar todas las páginas requeridas.

El escenario básico del manejo de los requerimientos de páginas se muestra en la figura 4-21.

La aplicación de la Tienda Virtual Fashion Pol es altamente dependiente de discretos controladores para manejar el flujo navegacional y coordinar la ejecución de la lógica del negocio, la figura 4-22 muestran las páginas que soportan la funcionalidad de la tienda y la figura 4-23 las clases Java que proporcionan acceso a la lógica del negocio en cada página, por medio de la clase ShopController.

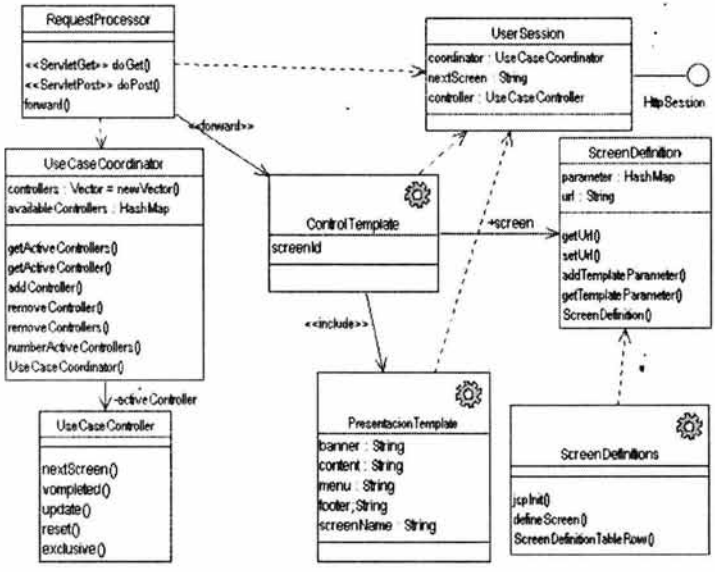

Figura 4-20: Clases del requerimiento de páginas.

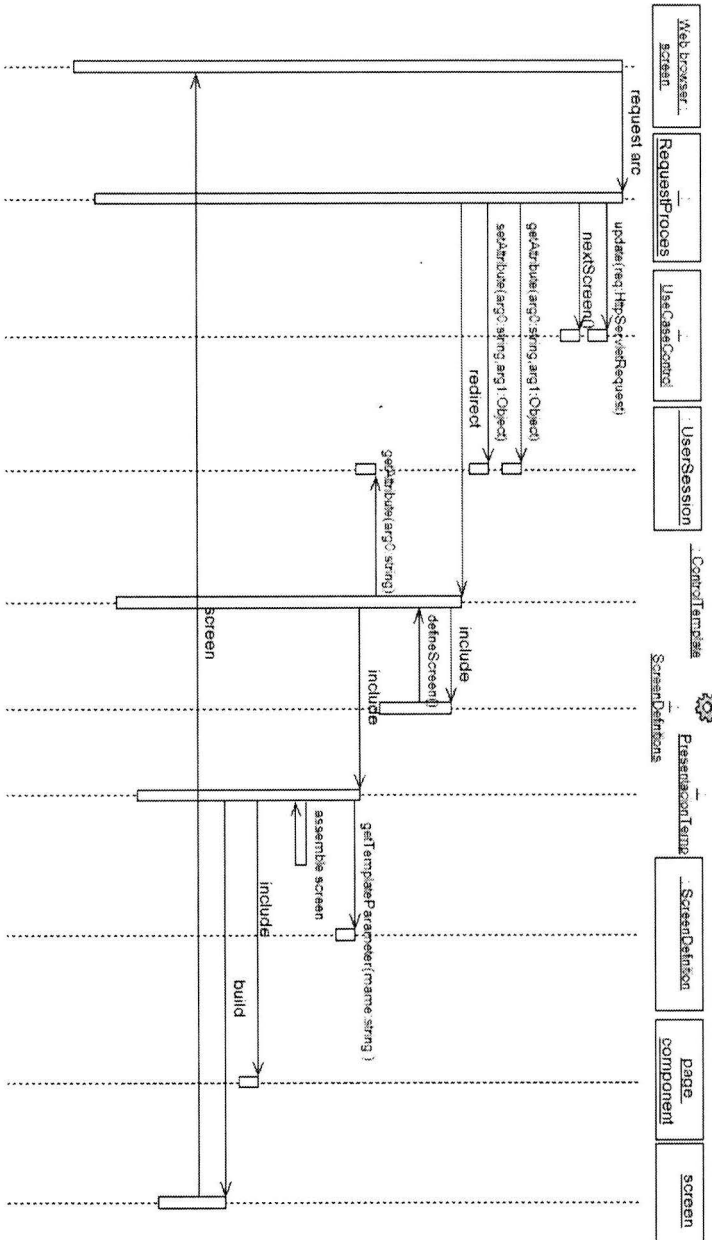

Figura 4-21: Escenario principal para atender requerimientos de páginas.

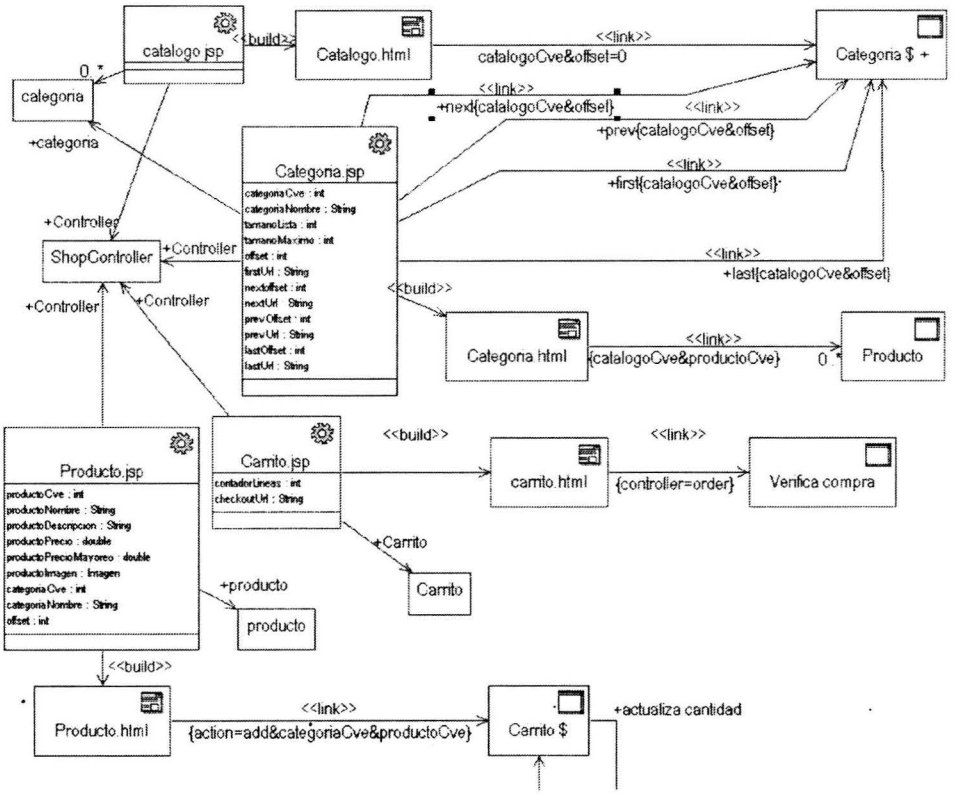

Figura 4-22: Páginas soportadas por la funcionalidad de la tienda.

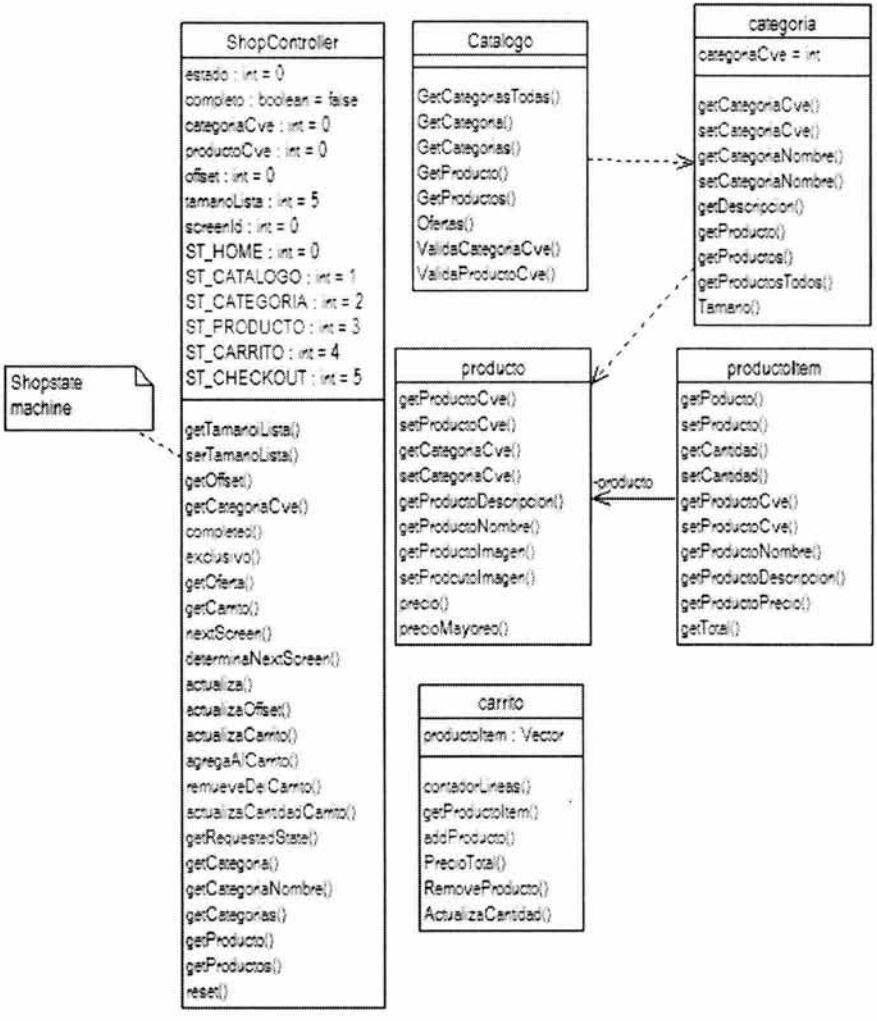

Figura 4-23
Clases Java involucradas en la funcionalidad.

De esta forma el conjunto de controladores de la aplicación definen una máquina de estados, la cual ayuda a definir operaciones en términos de la siguiente pantalla a presentar.

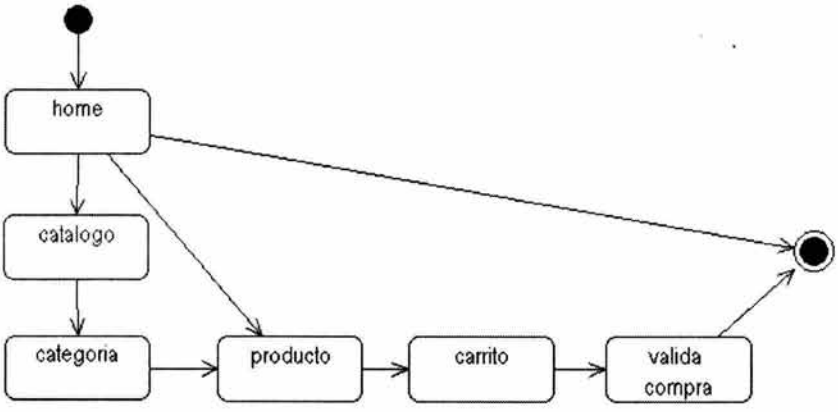

Figura 4-24: Diagrama de estados para los controladores.

4.3.8 Vista de componentes.

La vista de componentes del modelo consta de dos jerarquías diferentes, pero conectadas entres sí, la primera identificada por el paquete estereotipo <<virtual root>> que contiene componentes que mapean directamente a elementos URLs validos para que el sistema pueda responder a los requerimientos, donde cada elemento es un estereotipo <<HttpResource>> y realiza los elementos screen del modelo UX. El segundo paquete estereotipo es <<phisical root>> contiene estereotipos de elementos JSP y HTML que están asignados a archivos JSP y HTML durante la ejecución del sistema de archivos.

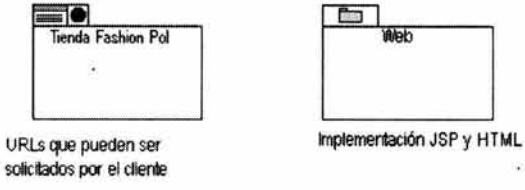

Figura 4-25: Diagrama de componentes de primer nivel.

4.4 Evaluación

La extensión WAE es un conjunto de especificaciones para modelar aplicaciones WEB centradas en la arquitectura; sin embargo, no todo lo propuesto por la metodología de Conallen es aplicable en todos los casos, es por ello importante, determinar en una fase temprana del diseño las consideraciones arquitectónicas que son relevantes para la aplicación a modelar.

WAE concentra su esfuerzo de diseño en tres niveles de las aplicaciones WEB, el primero es un nivel conceptual que esta basado en una relación Orientada a Objetos, un segundo nivel estructural, que se representa por medio de enlaces entre componentes y nodos, y en el tercer nivel, un nivel visible que incluye un conjunto de marcos, formularios, elementos de entrada, de texto y de selección. Es un rico conjunto de elementos que extienden la notación UML para cubrir el modelado de aplicaciones WEB detallando los aspectos en ambos lados del esquema cliente/servidor.

Finalmente, la extensión WAE, puede considerarse una característica avanzada de UML, por lo cual es importante tener experiencia empleando el Lenguaje Unificado de Modelado antes de iniciar con el uso de la extensión, es importante iniciar su aprendizaje en proyectos pequeños, ya que UML y la extensión están pensados para proyectos que incluyen gran cantidad de software y que involucran a diversos equipos de trabajo, es por esto, que considero que la extensión WAE es una buena herramienta para el diseño de aplicaciones WEB, ya que permite modelar la aplicación completa, con los aspectos de una aplicación cliente/servidor y todos los elementos de las aplicaciones WEB, sin embargo, no hay que olvidar que solo una

herramienta, y su aprovechamiento depende de la experiencia que tenga el equipo de desarrollo en su aplicación.

En relación con el trabajo realizado se puede decir, que a pesar de mostrar un modelo que no está completamente refinado, si fue posible probar algunas de las características de UML y su extensión WAE para el diseño de aplicaciones WEB, y evaluarla como una herramienta que con algo de experiencia en su uso, puede ayudar a los diseñadores de aplicaciones WEB a cumplir con las actividades de la etapa del diseño del software.

4.4.1 Ejemplos de pantallas para el módulo de Clientes.

Del modelo realizado para la aplicación WEB Fashion Pol, se deduce que el tipo de cosas que el usuario desea ver en algún punto de la aplicación son similares a las que se observan en las figuras 4-26 a 4-29.

Figura 4-26: Pantalla Home.

Figura 4-27: Pantalla Acerca de.

Figura 4-28: Pantalla Productos.

Figura 4-29: Pantalla carrito de compras.

Como puede observarse, las pantallas muestran la satisfacción de los requerimientos de navegación descritos en la fase del análisis del problema.

CONCLUSIONES Y RECOMENDACIONES.

- UML surgió como un lenguaje de modelado genérico; adaptable, al menos en principio a cualquier tipo de aplicación, para lo cual proporciona mecanismo de extensibilidad que permiten adaptar el lenguaje. En poco tiempo UML se ha convertido en el lenguaje de modelado más usado; convirtiéndose, de hecho en un estándar, como resultado de un proceso de mejoramiento en el que han contribuido sus creadores originales, así como numerosas organizaciones.

- Uno de los aspectos en que se trabajo para extender este lenguaje es el campo de las aplicaciones WEB, respondiendo a la gran aceptación que ha tenido el Internet, y con ello el crecimiento del número y tipo de aplicaciones de este tipo. La extensión WAE pretende cubrir las carencias que para este tipo de aplicaciones tiene UML.

- Este trabajo tuvo como finalidad la construcción de un modelo para una aplicación de comercio electrónico con un lenguaje único, siendo este el lenguaje UML y su extensión de notación WAE, objetivo que se puede dar por cumplido, ya que se presentan los diagramas (planos) necesarios para construir el software de la aplicación de tienda virtual Fashion Pol, a pesar de que estos se presentan en un nivel

alto de abstracción, son suficientes para mostrar los beneficios que ofrece UML y su extensión WAE, además no son un modelo que este concluido, ya que este puede ser ampliado para obtener un mayor detalle de la aplicación.

- En general, no existe una regla que nos indique en que casos es factible para una empresa colocar una sucursal en línea, no es el tamaño de la organización lo que determina el éxito de un proyecto, el presente estudio fue orientado a una empresa mediana dedicada a la venta de ropa, pero puede generalizarse otras empresas medianas o pequeñas de cualquier giro.

- De la experiencia de este trabajo puedo asegurar que a pesar de no existir un estándar para el diseño de aplicaciones WEB, el trabajo que se ha desarrollado para el Lenguaje Unificado de Modelado y también para la extensión WAE, se dirige a conseguir mayor aceptación de estas herramientas para la construcción de planos de software, para cualquier tipo de aplicación, en particular para las que se ejecutan en ambiente WEB, ayudan a mantener una buena comunicación entre el cliente del software y el equipo que se encuentra trabajando en el proyecto de software.

- La extensión WAE, no es un lenguaje de programación, basta recordar que modelar no es programar y programar no es modelar; WAE, es una extensión de la notación del Lenguaje Unificado de Modelado; por lo tanto no es, ni pretender ser en ningún momento una herramienta única para la construcción de una aplicación WEB, sin embargo es muy

útil a la hora de construir planos para la construcción de aplicaciones.

- Adicional al uso de esta o cualquier otra herramienta, deben tomarse en cuenta algunas recomendaciones:

  √ Para construir aplicaciones Web, es importante definir al principio del proyecto, algunos aspectos que serán factor de éxito de la aplicación, por ejemplo determinar el número de capas en que una aplicación del tipo cliente/servidor deberá diseñarse de acuerdo a las características propias del problema concreto a resolver considerando lo siguiente:

  Dos capas son suficientes cuando: El usuario es libre de navegar por las páginas de la aplicación en cualquier orden, y el número de páginas y objetos es relativamente pequeño, y las transacciones no requieren modificar la base de datos.

  Por otro lado se requiere un sistema en tres capas cuando: Existe una relación compleja entre la base de datos y los objetos de la aplicación, las transacciones requieren actualizar las bases de datos, y se puede considerar el sistema complejo por incluir un número relativamente grande de páginas y objetos, y la navegación del usuario es determinada por la página previa visitada.

  √ Es importante, en una aplicación WEB, separar la lógica del negocio, de la presentación de la aplicación.

- √ Determinar el tipo de seguridad que la aplicación requiere y la forma en que esta será implementada.

- √ Diseñar la aplicación pensado en lo que un cliente desea ver y encontrar. Manteniendo siempre una buena comunicación con los clientes de la aplicación, y atendiendo a las recomendaciones de color, estructura de las páginas, contenido e imágenes a incluir, optimizando los recursos y facilitando la navegación de la aplicación por parte del usuario.

- √ Para el diseño de las aplicaciones WEB un modelo importante a generar es el de experiencia del usuario (UX), que en su primera versión deberá ser modelo general, que no deberá pensarse para la creación de código, y puede ser utilizando para identificar contenidos importantes, nombres de campos y formularios de entrada.

- √ Puede iniciarse el modelo de la aplicación utilizando las clases estereotipadas de <<server page>>, <<client page>>, y <<form>> solamente. También pueden combinarse estos elementos con otros del modelo general sin poner atención a la asignación de dirección URL a las páginas.

- Una ventaja de un modelo diseñado con UML, es la facilidad con que el cliente del proyecto puede involucrarse en el proceso del diseño del software y observar la evolución de su producto verificando el cumplimiento de los requisitos del modelo.

- Se puede decir al final del presente trabajo, que el objetivo planteado al inicio del mismo, es decir, la construcción de un modelo para una aplicación WEB; se cumple, dado que al final se presenta el diseño, que puede ser utilizado para construir la aplicación. Si bien es cierto que actualmente existen muchos productos que nos ofrecen, con cierta facilidad, colocar productos en Internet, lo cierto es que una aplicación general no puede cubrir satisfactoriamente todos los requisitos de una empresa en particular, es por ello, que considero que un estudio como éste puede ayudar a las empresas a cubrir de forma mas amplia sus requisitos.

- Este proyecto plantea un problema que requiere de comunicación con el usuario, para su completa comprensión, y a su vez concluye con un modelo que tiene como objetivo resolver esa problemática; por ello pienso que refuerza al profesional de Matemáticas Aplicadas y Computación como aquel que tiene la capacidad de interactuar con personas de otros ámbitos y ayudarle a resolver problemas utilizando la computadora y las herramientas de software, en este caso Rational Rose, UML y la extensión WAE.

BIBLIOGRAFÍA.

Booch, Grady, **Análisis y diseño orientado a objetos con aplicaciones,** Ed. Addison Wesley Longman, México, 1998.

Booch, Grady. Rumbaugh, James. Jacobson, Ivar, **El lenguaje unificado de modelado,** Ed. Addison Wesley, España, 1999.

Conallen, Jim, **Building WEB Applications with UML Second Edition,** Ed. Addison Wesley, EUA, 2002.

Elsenpeter, Robert C. Velte, Toby J, **Fundamentos de Comercio electrónico,** Ed Osborne McGraw-Hill, México, 2001.

Parker, Timothy, **Aprendiendo TCP/IP en 14 días**, Ed. Prentice Hall, México, 1996.

Pressman, Roger S, **Ingeniería de Software, Un enfoque práctico, cuarta edición,** Ed. Mc Graw Hill, México, 2000.

Yourdon, Edward, **Análisis estructurado moderno,** Ed. Prentice Hall, México, 1993.

Stevens Perdita, **Utilización de UML en Ingeniería del Software con Objetos y Componentes,** Ed. Addison Wesley, España, 2002.

4.5 Libros en línea:

Putman, Janis R, **Architecting with RM-ODP,** Ed Prentice Hall PTR, EUA, 2003

Sawyer, Ben. Greely, Dave. Cataudella, Joe, **Creating Stores on the WEB, 2nd Edition,** Peachpit Press, EUA, 2000

Walther, Stephen. Levine, Jonathan, **Sam´s Teach Yourself E-Commerce Programming with ASP in 21 Days,** Ed. Sam´s, EUA, 2000.

4.6 Documentos en Internet:

Administración Ventas, [en línea], Adriana Rodríguez Martínez, [citado el 27/mar/2003], disponible en: http://www.toditopersonal.com/sitios/adriana_ecomerce/pres.asp

Algunos desafíos para el desarrollo del comercio electrónico de la América Latina de habla hispana, [en línea], Charles Davis, publicado en Mar-98, [citado el 01/mar/2003], Conferencia ofrecida en el VII seminario Latinoamericano de gestión Tecnológica ALTEC 97, disponible en: http://www.idrc.ca/lacro/docs/conferencias/panamericas.html

Ciclo de vida del software, [en línea], [citado el 25/abr/2003], disponible en: http://www.inf_cr_uclm.es/www/cbravo/is/tema02.pdf

Comercio electrónica, [en línea], Vanessa Corona Torres, modificado el 06-May-02, [citado el 13/mar/2003], disponible en: http://mailweb.udlap.mx/~104418

Conceptos Básicos de World Wide WEB, [en línea], Unidad de redes y Comunicaciones de la Universidad de Jaén, publicado el 04-Feb-99, [citado el 21/mar/2003], disponible en: http://www.ujaen.es

Cronología de Internet de Hobbes v5.4, [en línea], Robert H'obbes' Zakon, modificado en Dic-02, [citado el 20/mar/2003], disponible en: http://isoc.org/zakon/Internet/History/HIT.html

Desarrollo de Aplicaciones WEB, [en línea], Álvaro López Ortega, publicado el 01-Mar-02, [citado el 12/mar/2003], disponible en: http://es.tldp.org/Presentaciones/200203cecic-mexico/conf-alo/html/cecic2002.html

Designing E-Business WEB Sites -Part 2 of 2: Designing for a Specific Audience, [en línea], Meghraj Thakkar, publicado el 17-Mar-03, [citado el 03/mar/2003], disponible en: http://www.informit.com/isapi/product_id

Enterprise Modelling News, [en línea], R.T.S., [citado el 05/jun/2003], disponible en: http://www.rts.nl/Entnews2-201.htm

Ingeniería del Software (Use cases), [en línea], Javier Martínez de Ibarreta L., publicado el 01-Ene-01, modificado el 01-Nov-01, [citado el 21/ene/2003], disponible en: http://www.euskalnet.net/javierml/softeng/usecases.htm

Programación Modular, [en línea], J.M. Ruiz M.,modificado en Ene-03, [citado el 05/abr/2003], disponible en: http://www.abacusnt.com/tem26.pdf

Setting Up Web Site Categories, [en línea], Lynda Weinman, publicado el 14-Feb-03, [citado el 20/mar/2003], InformIT.com:Articles>Setting Up Web Site Categories, disponible en: InformIT.com:Articles>Setting Up Web Site Categories

The Future of WEB of Services Security: A Conversation with Eve Maler, [en línea], Janice J. Heiss, publicado el 18-Mar-03, [citado el 24/mar/2003], disponible en: http://www.java.sun.com/features/2003/03/webservices-qa.html

The web modeler, [en línea], Jim Conallen, publicado el 01-Mar-2000, [citado el 05/mar/2003], disponible en: http://www.therationaledge.com/rosearchitect/mag/current/spring00/webmod.html

Una breve historia de Internet, [en línea], Barry M. Leiner, [citado el 20/mar/2003], disponible en: http://www..aui.es/historia/ihistoria.htm