UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

DESARROLLO DE UN SISTEMA PARA EL CONTROL Y SEGUIMIENTO DE ÓRDENES Y PRODUCTOS CON TECNOLOGÍA WEB

TRABAJO ESCRITO

EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:
PAOLA GARCÍA CAMPOS

ASESOR: MTRO. JESÚS HERNÁNDEZ CABRERA

MÉXICO **2007**

Índice

Introducción.

En la construcción y desarrollo de proyectos se aplican métodos y técnicas para resolver problemas, la informática aporta herramientas y sobre los que se apoya la ingeniería del software.

Las organizaciones almacenan grandes cantidades de datos, razón por la cual debe tenerse en cuenta dónde almacenarlos y la manera de recuperarlos en el momento en que se necesita. Cuando un sistema se desarrolla de manera adecuada, será posible recuperar de manera rápida dicha información.

Los factores que deben tomarse en cuenta para un sistema de información son por ejemplo, la tecnología de comunicaciones a utilizar, el impacto que tendrá dicho sistema sobre los empleados de la empresa y que las características específicas que debe tener el sistema se determinen de manera secuencial.

El proceso de desarrollo de Software requiere de un conjunto de conceptos, una metodología y un lenguaje propio. Actualmente el desarrollo de software se encuentra en una etapa en la cual se enfoca a la Orientación a Objetos.

Existen conceptos ligados en torno a la tecnología orientada a objetos, los principios, el análisis y el diseño, todos éstos se explican en los primeros capítulos de este trabajo, ya que es importante saberlos para poder desarrollar el software de la mejor manera.

En este trabajo se desarrolla un pequeño sistema en el cual se aplican los conocimientos adquiridos en el diplomado Desarrollo de Sistemas en Web. Intento plasmar paso a paso el desarrollo desde la problemática, el análisis del sistema mediante diagramas UML, el diseño y finalmente una parte del desarrollo del sistema siendo lo más explícita posible.

Los conceptos que tendremos en el capítulo 1 y 2, son de mucha utilidad, ya que con ellos tenemos las bases necesarias, comenzando por dar una breve historia y los servicios existentes de Internet, el concepto de redes, los conceptos básicos de la programación orientada a objetos, una definición y descripción de lo que es el lenguaje JAVA, la descripción de lo que es UML así como los diagramas que en este se utilizan.

En el capítulo 3 se explica más detalladamente lo que es el Lenguaje Unificado para el Modelado de Software (UML), esto servirá para lograr una mayor comprensión de los diagramas que se exponen más adelante.

En el capítulo 4 se analiza el sistema comenzando con el enunciado general, la problemática, los requerimientos funcionales y no funcionales y, utilizando UML, se exponen los diagramas de casos de uso, casos de uso formato expandido y diagramas de análisis con una breve explicación de cada uno de ellos.

El capítulo 5 es acerca del diseño del sistema, el cual contiene diagramas de secuencia a bajo y alto nivel y los diagramas de secuencia, que son de mucha utilidad para definir cómo va a funcionar el sistema.

Y por último, en el capítulo 6, que lleva por nombre Implementación y pruebas, se muestran y explican las pantallas de las que comprende el sistema y el código de un módulo del sistema.

1. Conceptos básicos para el desarrollo de aplicaciones en WEB.

En este capítulo se encuentran algunos de los conceptos básicos para el desarrollo de aplicaciones en web, comenzando con la historia y el concepto de Internet, su estructura, tipos de servicios de Internet, entre otros.

1.1 Internet

Internet es un sistema mundial integrado por las diferentes redes de cada país del mundo, por medio del cual un usuario en cualquier computadora puede tener acceso a información de otra computadora en caso de contar con los permisos necesarios e inclusive entablar comunicación directa con otros usuarios en otro lugar del mundo.

1.1.1 Historia de Internet.

Internet fue desarrollado por una agencia de nombre ARPA (Advanced Research Projects Agency) del gobierno de los Estados Unidos en a principios de la década de los 70's y se le conocía inicialmente como ARPANET. El Departamento de Defensa tenía como propósito original crear una red que permitiera seguir en contacto en caso de un desastre. Otra condición era que si parte de la red era dañada o destruida, el resto del sistema debía seguir funcionando. Esta red puso por primera vez en contacto a los investigadores y científicos académicos estadounidenses. Fue, además, la predecesora de Internet que conocemos actualmente.

En 1985 y basada en los protocolos de comunicación de ARPANET, la Nacional Science Foundation (NSF) creó NSFNET, una red nacional diseñada para difundir nuevos descubrimientos, ofreciéndola a instituciones americanas dedicadas a la educación e investigación, uniéndose a ella con el fin de enlazar el tráfico electrónico de instituciones individuales a otras redes regionales que fueron apareciendo.

Rápidamente creció la NSFNET, a la par con el descubrimiento del público de su potencial y con el surgimiento de nuevas aplicaciones que facilitaban su acceso. Corporaciones como Sprint y MCI empezaron a construir sus propias redes, que enlazaron con NSFNET. Mientras firmas comerciales y otros proveedores de red regionales empezaron a hacerse cargo de las operaciones de las mayores arterias de Internet, NSF ha ido dejando de dar soporte al esqueleto de la red.

Fue entonces cuando se empezó a extender Internet por los demás países del mundo, enlazando la comunicación entre Europa y EE.UU.
Actualmente, el Internet es un medio de comunicación público, cooperativo y autosuficiente en términos económicos, es muy accesible a cientos de millones de gente en el mundo. El Internet usa parte del total de recursos actualmente existentes en las redes de telecomunicaciones. Técnicamente, lo que distingue al Internet es el uso del protocolo de comunicación llamado TCP/IP (Transmission Control Protocol/Internet Protocol).

1.1.2 Concepto de Internet.

"La red Internet es el resultado de comunicar miles de redes de computadoras entre sí. Permite conectar diferentes tipos de redes, que pueden ser de área local o de área extensa".[1]

Por medio de esta enorme red se envía continuamente innumerables cantidades de información. Millones de personas navegan por Internet en todo el mundo. Se le llama navegar porque es usual el hallar información proveniente de muchas partes distintas del mundo en una sola sesión.

Una de las ventajas de Internet es que hoy en día podemos realizar una conexión con cualquier tipo de computadoras, desde las personales, hasta las más grandes y hacer diferentes cosas, como por ejemplo, enviar y revisar correos electrónicos, anunciar algún producto o servicio, leer noticias, bajar información de interés, ya que hay bibliotecas con millones de libros y lecturas ilimitadas, hacer compras, movimientos bancarios, etcétera.

1.1.3 Estructura de Internet.

Internet es una red que está por encima de las redes de tipo LAN o WAN, que eran redes que unían computadoras de empresas o de particulares.

Actualmente existe gran variedad de lenguajes que usan las computadoras para poder contar con una conexión a Internet. Estos lenguajes son llamados Protocolos. Se ha establecido que en Internet, toda la información debe de ser transmitida mediante el Protocolo TCP/IP.

El funcionamiento de Internet se lleva a cabo mediante la estrategia Cliente/Servidor, o sea que en la red hay computadoras que se denominan Servidores y su funcionamiento es dar una información concreta en el momento que se solicita, y por otro lado, también existen las computadoras denominadas Clientes, que son las que solicitan la información.

1.1.4 Protocolo TCP/IP.

TCP/IP (Transfer Control Protocol/Internet Protocol) es el lenguaje establecido por Internet y es utilizado por todas las computadoras que se enlazan a Internet con el fin de que puedan comunicarse entre sí.

En Internet se encuentran conectados variados tipos de computadoras que pueden ser incompatibles entre sí, debido a esto, el protocolo TCP/IP se encarga de que la comunicación entre todas las máquinas sea posible, ya que es compatible con todo tipo de hardware y sistema operativo.

[1] Ferreira C. Gonzalo, Internet paso a paso. Hacia la autopista de la información, Alfaomega Gpo. Editor, S.A. de C.V., México 1996, p.34

La arquitectura TCP/IP es actualmente muy difundida y se divide en 4 niveles:

1. *Nivel de subred*: (Enlace y físico) Análogo al nivel físico del OSI (Open Systems Interconnection).

2. *El nivel de interred.* (Red, IP) Este nivel Incluye al protocolo IP con la finalidad de los protocolos de nivel de transporte, es el responsable de enviar los paquetes de información a su destino. Es el nivel de red del modelo OSI.

3. *El protocolo proveedor de servicio.* Los protocolos de este nivel, (tales como TCP y UDP), son responsables de manejar y dar la fiabilidad necesaria en el transporte de los datos. Coincide con el nivel de transporte del modelo OSI.

4. El nivel de aplicación incluye protocolos destinados a proporcionar servicios, como conexión remota, transferencia de ficheros (FTP), correo electrónico y protocolo HTTP (*Hypertext Transfer Protocol*) entre otros. Este nivel corresponde con los niveles OSI de aplicación, presentación y sesión.

TCP/IP es la base para muchos servicios útiles. El protocolo IP es parte integral del TCP/IP y sus principales responsabilidades son el direccionamiento de los datagramas de información, así como la administración de su proceso de fragmentación.

Las características del protocolo IP son:

La unidad de transferencia que el IP utiliza es por medio de datagramas, denominadas en forma más específica como datagramas Internet o datagramas IP. No tiene corrección de errores, ni control de congestión y no está garantizada la entrega en secuencia del datagrama, ya que es posible que se retrase o encamine de modo incorrecto o puede enviarse incompleto al dividir los fragmentos del mensaje.

En cuanto al ruteo puede ser mediante tablas de rutas estáticas o dinámicas, direccionamiento IP o paso a paso a todos los nodos.

El TCP/IP utiliza una dirección de 32 bits para identificar una máquina y la red a la cual está conectada. Únicamente el NIC (Centro de Información de Red) asigna las direcciones IP, aunque si una red no está conectada a Internet, dicha red puede determinar su propio sistema de numeración.

Actualmente existen cuatro formatos para la dirección IP, cada uno de los cuales se utiliza dependiendo del tamaño de la red, como se muestra en la figura 1.1, son a partir de la Clase A hasta Clase D (para un futuro se tiene pensado añadir la Clase E).

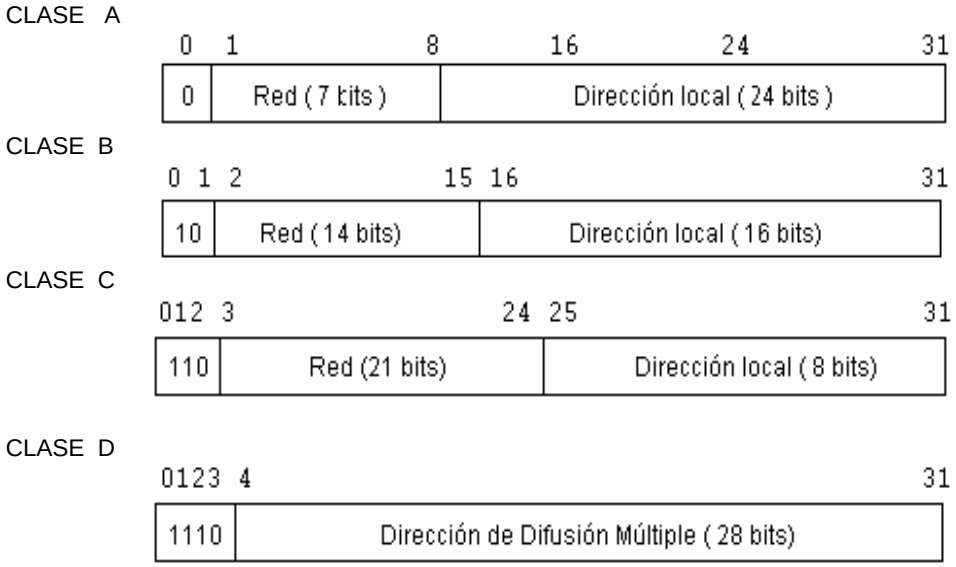


Figura 1.1. Formatos para determinar la dirección IP

No es posible que coexistan dos computadoras diferentes que tengan la misma dirección por una sencilla razón, la información solicitada por cualquiera de las dos máquinas no sabría hacia dónde ir.

Cada dirección está compuesta por un par (RED (netid), y Dir. Local (hostid)) en donde se identifica la red y el host dentro de la red.

La clase se identifica mediante las primeras secuencias de bits, a partir de los 3 primeros bits (de orden más alto).

Las direcciones de Clase A pertenecen a grandes redes con muchas máquinas. Las direcciones en decimal son 0.1.0.0 hasta la 126.0.0.0 (lo que permite hasta 1.6 millones de hosts).

El formato de Clase B corresponde a redes de tamaño intermedio, el rango de direcciones varía desde el 128.0.0.0 hasta el 191.255.0.0. Permitiendo tener 16320 redes con 65024 host en cada una.

La Clase C consta de sólo 8 bits para la dirección local (host) y 21 bits para red. Las direcciones de esta clase comprenden entre 192.0.1.0 y 223.255.255.0, permitiendo cerca de 2 millones de redes con 254 hosts cada una.

La Clase D es útil con fines de multidifusión, en caso de necesitar una difusión general a más de un dispositivo. El rango es desde 224.0.0.0 hasta 239.255.235.255.

Aunque su utilización será futura, la clase E comprende el rango desde 240.0.0.0 hasta el 247.255.255.255.

La dirección IP consta de 4 conjuntos de 8 bits, con un total de 32 bits representados como si se encontraran separados por un punto. El formato de dirección IP puede ser re.local.local.local para Clase A hasta red.red.red.local para la clase C.

La distribución jerárquica de Internet permite enviar y recibir rápidamente paquetes de información entre dos computadoras que estén conectadas a Internet en cualquier parte del mundo.

Las direcciones IP son manejadas por las computadoras mediante el protocolo TCP/IP de manera invisible, por lo que el usuario no necesita saberlas ya que para pedir información a una computadora utiliza el Nombre del Dominio, que es la dirección IP traducida para el usuario.

Las computadoras que cuentan con Nombre de Dominio son los servidores, ya que son los que reciben solicitudes de información, una computadora cliente que sólo solicita información no necesita un nombre de dominio, ya que ninguna computadora le va a solicitar ninguna información.

1.1.5 FTP (File Transfer Protocol o Protocolo de Transferencia de Archivos).

El FTP (File Transfer Protocol) es un sistema que permite la transferencia de archivos, con el objetivo de obtener o copiar archivos de una computadora a otra remota de la red. Mediante este servicio, Internet permite el acceso a bases de datos que contienen software de dominio público e información sobre diversos temas.

Para copiar archivos desde una computadora a otra, es necesario tener una cuenta en el host remoto y un password para lograr el acceso a dicha cuenta. FTP hace una conexión especial con el host remoto, lo que nos permitirá navegar en los directorios y seleccionar los archivos que se van a transferir.

FTP permite el acceso a archivos almacenados en computadoras centrales desde computadoras personales, distribuye información a través de la red y permite el acceso a bases de datos públicas. Su funcionamiento se basa en un conjunto de comandos que permiten el acceso a la información de una maquina remota.

1.2 Servicios de Internet.

En este apartado se citan algunos de los servicios de Internet y en qué consisten.

1.2.1 Definición de Servicios de Internet.

Estos servicios son las posibilidades que ofrece Internet con el fin de satisfacer las necesidades de los usuarios. Cada servicio es un modo de aprovechar esta tecnología y es importante que con este fin, el usuario conozca la mayoría de los servicios que ofrece.

1.2.2 Tipos de Servicios de Internet

En estos días, los servicios más comunes son los siguientes: Correo Electrónico, World Wide Web, FTP, Grupos de Noticias, IRC y Servicios de Telefonía.

Correo Electrónico: El correo electrónico fue una de las primeras aplicaciones creada para Internet y es el servicio de envío y recepción de mensajes de texto. Prácticamente ha reemplazado al servicio postal para algunos usuarios en caso de breves mensajes escritos, debido a su rapidez y a que es más barato enviar un e-mail que una carta postal. Además en caso de que ocurriera algún problema con el servidor, los mensajes que no alcanzaron a enviarse se retienen y esperan a que se restablezca algún punto para que lleguen a su destino.

World Wide Web: Es el universo de información accesible a través de Internet, Una de sus características es que tiene el texto remarcado, que es un método para referencias cruzadas instantáneas. En la mayoría de los Sitios Web, ciertas palabras aparecen en texto de otro color diferente al resto del documento. Por lo general, este texto es subrayado.

Mediante el uso del Web se puede tener acceso a muchísima información, y para poder realizar una exploración en el Web, puede ser mediante un software denominado Bowser o Explorador.

Grupos de Noticias: Es un servicio de gran utilidad para resolver algunas dudas. En estos grupos se reúne la gente para tratar de intereses comunes. En los grupos de noticias se puede encontrar discusiones sobre política y religión, fotografías, anuncios sobre trabajos, consejos sobre negocios y salud, etcétera.

IRC: El servicio IRC (Internet Relay Chat) nos permite entablar una conversación en tiempo real con una o varias personas de cualquier parte del mundo por medio de texto y enviar imágenes u otro tipo de archivos mientras se dialoga.

1.2.3 Secure Shell

SSH (Secure SHell) es un software que permite realizar una conexión remota más segura a través de canales inseguros a sistemas, es utilizado para llevar a cabo la transferencia de archivos desde o hacia un sistema remoto de manera segura. Con SSH se inicia una conexión de una máquina cliente con una máquina servidor. SSH facilita la tarea de cifrar diferentes tipos de comunicación que normalmente se envía en modo inseguro a través de redes públicas. Los tipos de protección que proporciona son los siguientes:

Todos los datos enviados y recibidos durante la conexión se transfieren por medio de una encriptación fuerte, lo cual los hacen extremamente difícil de descifrar y leer.

Después de realizar una conexión inicial, el cliente tiene la posibilidad de revisar que se está conectando al mismo servidor en sesiones anteriores como en esta.

El cliente puede transmitir su información de autentificación al servidor, como el nombre de usuario y la contraseña, en formato cifrado.

El cliente tiene la posibilidad de usar X11 aplicaciones lanzadas desde el indicador de comandos de la shell. Esta técnica proporciona una interfaz gráfica segura (llamada reenvío por X11).

1.2.4 Protocolo HTTP

HTTP (HyperText Transfer Protocol) es un protocolo de transacción de la web donde el hipertexto es el contenido de estas y el protocolo de transferencia es el sistema por medio del cual se envían las peticiones de acceso a una pàgina web y la respuesta de esa web proporcionando la información que se verá en pantalla. Este protocolo no tiene estado, ya que no tiene la capacidad de guardar ninguna información de conexiones pasadas y cuando se finaliza la transacción se pierden los datos. Por este motivo se popularizaron las *cookies*, que son ficheros que se guardan en la computadora para leer un sitio web cuando se establece la conexión con él y poder reconocer a un visitante que ya tuvo conexión con este sitio en ocasiones anteriores.

El protocolo HTTP está basado en el modelo cliente-servidor, donde un cliente abre una conexión y solicita algo al servidor, el cual responde a dicha solicitud y finaliza la conexión.

1.2.5 MySQL

“MySQL: Sistema gestor de base de datos relacional cliente-servidor de coste mínimo que incluye un servidor SQL, programa cliente para acceder al servidor, herramientas administrativas y una interfaz de programación para escribir programas”.[2]

MySQL es un Sistema de gestión de Base de Datos Relacional, que se caracteriza por disponer que toda la información debe estar almacenada en tablas, las relaciones entre los datos se deben representar en esos mísmos datos y además MySQL añade velocidad y una gran flexibilidad, así como confiabilidad y facilidad para usarse en volúmenes tanto pequeños como grandes.. En MySQL podemos agregar, acceder, y procesar datos grabados en una base de datos.

Tenemos la posibilidad de descargar MySQL de Internet, ya que utiliza el GPL que es una Licencia Pública General y es de manera gratuita, por lo que se puede estudiar su código y cambiarlo de acuerdo a su conveniencia.

[2] Pérez César, MySQL para Windows y Linux, Alfaomega, Gpo. Editor, S.A. de C.V., México D.F. 2004, p. 1

IBM empezó a comercializar en 1981 el SQL y desde entonces este producto ha tenido un papel importante en el desarrollo de la bases de datos relacionales.

En 1983 nació DB2, la más popular en los grandes ordenadores de las bases de datos de este tipo hasta este momento.

Alrededor de la década del 90, Michael Windenis comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo. Con el tiempo,llego a la conclusión que mSQL no era tan rápido ni flexible para sus necesidades. De esto surgió en una nueva interfaz SQL con código más portable. Esto había sido hecho para lograr con relativa facilidad portar aplicaciones y utilidades de MiniSQL a MySQL.

- Características de MySQL

- El principal objetivo de MySQL es velocidad y robustez.
- Clientes C, C++, JAVA, Perl, TCL.
- Puede usar varias computadoras si estas están disponibles.
- Puede trabajar en distintas plataformas y Sistemas Operativos distintos.
- Sistema de contraseñas y privilegios muy flexibles y seguros.
- Todas las palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- Todas las columnas pueden tener valores por defecto.
- Utilidad para optimizar y reparar tablas.
- Todos los datos están grabados en formato ISO8859_1.
- Los clientes usan TCP o UNIX Socket para conectarse al servidor.
- El servidor soporta mensajes de error en distintas lenguas.
- Todos los comandos tienen -help o -? Para las ayudas.
- Diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, etc.
- ODBC para Windows 95 (con fuentes), se puede utilizar ACCESS para conectar con el servidor.

1.3 Redes

Una “red de computadoras”[3] es un conjunto de computadoras y/o dispositivos conectados por enlaces de un medio físico ó inalámbrico y que comparten información, recursos (CD-ROM, impresoras, etc.) y servicios (e-mail, Chat, juegos), etc.

1.3.1 Concepto de Red.

Una red es un conjunto de nodos conectados entre sí que permiten el intercambio de información entre las computadoras.

[3] También llamada red de ordenadores o red informática. Ferreira C. Gonzalo, Internet paso a paso. Hacia la autopista de la información, Alfaomega Gpo. Editor, S.A. de C.V., México 1996, p.58

1.3.2 Tipos de Redes

Las redes se clasifican según su tamaño y su distribución lógica.

a) Clasificación según su tamaño

Las redes *PAN* (red de administración personal) son redes pequeñas, las cuales están conformadas por no más de 8 equipos, por ejemplo: café Internet.

CAN: Campus Area Network, Red de Area Campus. Una CAN es un conjunto de redes de LAN que se distribuyen dentro de un campus (industrias, universitario, maquilas, etcétera) y que pertenecen a una entidad en un área previamente delimitada en kilómetros. Las redes CAN utilizan tecnología como FDDI para lograr una conectividad mediante fibra óptica y espectro disperso.

Las redes LAN (Local Area Network, redes de área local) son redes pequeñas que se utilizan en una oficina o empresa, tienen dimensiones muy limitadas, y debido a esto son rápidas y tienen una administración reducida y las estaciones tienen la posibilidad de comunicarse unas con otras, son redes muy rápidas en las cuales cada estación se puede comunicar con el resto.

Este tipo de redes utilizan una tecnología de transmisión por medio de un cable coaxial al que están conectadas todas las computadoras, la velocidad en que operan es entre 10 y 100 Mbps, los canales son de los usuarios, las estaciones están cerca una de otra, aumenta la eficiencia y productividad de los trabajos de oficina, tienen una mínima tasa de error, incluso menos que las redes WAN, y su arquitectura permite compartir recursos.

Las redes WAN (Wide Area Network, redes de área extensa) son redes punto a punto que interconectan países y continentes. Al tener que recorrer una gran distancia sus velocidades son menores que en las LAN aunque son capaces de transportar una mayor cantidad de datos. El alcance es una gran área geográfica, como por ejemplo: una ciudad o un continente. Está formada por una vasta cantidad de computadoras interconectadas (llamadas hosts), por medio de subredes de comunicación o subredes pequeñas, con el fin de ejecutar aplicaciones, programas, etc.

Una red de área extensa *WAN* es un sistema de interconexión de equipos informáticos geográficamente dispersos, incluso en continentes distintos. Las líneas utilizadas para realizar esta interconexión suelen ser parte de las redes públicas de transmisión de datos.

Las redes *LAN* comúnmente, se conectan a redes WAN, con el objetivo de tener acceso a mejores servicios, como por ejemplo a Internet. Las redes WAN son mucho más complejas, porque deben enrutar correctamente toda la información proveniente de las redes conectadas a esta.

Una subred está formada por dos componentes:

-Líneas de transmisión: Se encargan de llevar los bits entre los host.

-Elementos interruptores (routers): Son computadoras especializadas usadas por dos o más líneas de transmisión. Para que un paquete llegue de un router a otro, generalmente debe pasar por routers intermedios, cada uno de estos lo recibe por una línea de entrada, lo almacena y cuando una línea de salida está libre, lo retransmite.

Redes Basadas en servidor. Las redes basadas en servidor pueden tener más de un servidor dependiendo del número de tráfico, números de periféricos, etcétera, son mejores para compartir una gran cantidad de recursos y datos. Un administrador supervisa tanto la seguridad, como la operación de la red.

Las redes MAN (Metropolitan Area Network) son de área metropolitana, por lo que su ubicación geográfica está determinada por ciudad o municipio. Tienen dos buses unidireccionales independientes uno del otro en cuanto a la transferencia de datos. Es prácticamente una versión de las redes LAN, ya que su tecnología es similar. Tienen una cobertura mayor a 4 Km El mecanismo para la resolución de conflictos en la transmisión de datos que usan las MANs, es DQDB.

El DQDB consiste en dos buses unidireccionales donde todas las estaciones se conectan entre sí y cada bus tiene su cabecera y un fin. Si una computadora quiere transmitir a otra y su ubicación es a la derecha, utiliza el bus de abajo, y si está a la izquierda utiliza el de arriba.

Redes Punto a Punto: En una red punto a punto la seguridad se dificulta, ya que la administración no está centralizada. En este tipo de redes cada computadora puede actuar como cliente y como servidor.

b) Clasificación según su distribución lógica

Una computadora cliente es aquélla que utiliza los servicios solicitando información de los servidores, mientras que un servidor es una máquina que ofrece dicha información o servicios a otras computadoras en Internet, como por ejemplo el correo electrónico, alguna página web, acceso a base de datos, etc.
Todas las redes deben cumplir con algunas características como pueden ser confiabilidad al transporte de datos, un buen procesamiento de información y dispositivos adecuados.
Las redes tienen diferentes usos de acuerdo a las diferentes necesidades del usuario, unos ejemplos podrían ser:
Compañías - centralizar datos.
Confiabilidad "transporte de datos".
Aumentar la disponibilidad de la información.
Comunicación entre personal de las mismas áreas.
Compartir recursos "periféricos, archivos, etc".
Ahorro de dinero.

c) Topologías

Topología Bus: Permite que todas las estaciones reciban la información que se transmite, una estación trasmite y todas las demás escuchan. Esta topología necesita menor cantidad de cables, además en caso de que ocurra algún problema en alguna estación, no afectará al resto de la red. Aunque si existe un solo canal de comunicación entre las estaciones de red y el canal falla, las demás quedan incomunicadas, este problema puede solucionarse si se pone un bus paralelo alterno para estos casos.

Actualmente hay dos mecanismos para resolver los problemas de transmisión de datos y son los siguientes:

-CSMA/CD: Todas las estaciones se consideran iguales, por este motivo compiten por el uso del canal de transmisión, cuando una estación requiere transmitir escucha el canal, en caso de que otra estación esté efectuando alguna transmisión, deberá esperar hasta que termine, si no se está utilizando el canal, se hace la transmisión y escucha posibles choques, si existe alguno espera un intervalo de tiempo y lo intenta nuevamente.

-Token (trama de datos) Bus: Se usa un token para que pase de estación en estación en forma cíclica, es decir, forma un anillo lógico. Sólo tiene derecho del bus para transmitir o recibir datos la estación que tiene el token por un tiempo determinado, luego el token pasa a otra estación a quien se le ceden los derechos de transmisión. Ninguna estación puede transmitir si no tiene el token, sólo escucha y espera su turno. De este modo se evitan las colisiones que tiene el mecanismo anterior.

Redes en Estrella: Es una de las principales topologías, en esta se une la red a un punto único con control centralizado.

Redes Bus en Estrella: Tiene el objetivo de facilitar la administración de red. La red es un bus cableado físicamente como la red en estrella por medio de concentradores.

Redes en Estrella Jerárquica: Su estructura es utilizada en la mayoría de las redes locales y se hace por medio de concentradores conectados en cascada con el fin de formar una red jerárquica.

Redes en Anillo: En este caso todas las estaciones están unidas con un solo cable formando un círculo por lo que las señales circulan en un solo sentido regenerándose en cada nodo teniendo como ventaja que los cuellos de botellas son poco frecuentes. La desventaja es que como sólo existe un canal de comunicación entre las estaciones de red y este o alguna estación tiene algún problema, todas las estaciones quedan afectadas. Este problema se puede resolver si se pone un canal alterno para casos de fallos.

Actualmente hay un mecanismo para resolver los problemas de transmisión de datos y son los siguientes:

Token Ring: La estación se conecta al anillo por una unidad de interfaz (RIU), cada RIU es responsable del control de los datos cuando pasan por ella, así como de regenerar la transmisión y pasarla a la estación siguiente. Cuando la dirección de cabecera de una transmisión indica que los datos son para una estación determinada, la unidad de interfaz los copia y pasa la información a la estación de trabajo conectada a la misma.

En el mecanismo Token Ring cada estación que tiene el token cambia su estado de desocupado a ocupado en caso de que quiera transmitir y agrega los datos atrás y lo pone en la red, de lo contrario el token pasa a la estación siguiente y cuando este pasa nuevamente a la estación que transmitió, saca los datos y lo pone en desocupado nuevamente y lo regresa la red. Este mecanismo se usa en redes de área local.

1.4 Tecnología Orientada a Objetos

Actualmente una de las formas más populares de programar es mediante la Tecnología Orientada a Objetos en el desarrollo de proyectos de software debido a sus grandes capacidades y ventajas frente a las antiguas formas de programar.

1.4.1 Surgimiento de la Tecnología Orientada a Objetos

La Tecnología Orientada a Objetos (TOO) tiene sus inicios en la década de los 60, y surge con la necesidad de describir y simular fenómenos como sistemas de comunicación, redes neuronales, sistemas administrativos, etcétera, y está fundamentada en el proceso de construcción y utilización de conocimientos, los objetos y las clases son los pasos más importantes en la búsqueda de un nuevo cambio que reemplace, esta vez, parte del esfuerzo que implica la organización y utilización del conocimiento.

En 1961, Krystin Nygaard tuvo la idea de desarrollar un lenguaje con el fin de lograr una descripción de sistema y una simulación programable, es cuando crea SIMULA I que proporcionaba facilidades y nuevas aplicaciones.

En 1967 se creó SIMULA 67 donde se implementaron por vez primera los conceptos que en adelante serían elementos centrales en los Lenguajes Orientados A Objetos como objeto, clase y herencia.

En 1970 se crea el Smalltalk, este fue el mayor desarrollo de los lenguajes orientado a objetos y el primer lenguaje exclusivamente creado para la programación orientada a objetos.

Anteriormente, la programación se realizaba mediante una serie de pasos consecutivos que tenía estructuras consecutivas y bifurcaciones, de modo que los lenguajes que se basaban en este modo de programación no ofrecían flexibilidad y surgían problemas complejos cuando los sistemas eran muy

grandes. Por este motivo surgieron lenguajes que ahora se basan en la programación estructurada, que tienen como objetivo principal descomponer las partes complejas del programa en módulos que se ejecutaba conforme se requiera, obteniendo un diseño compuesto por módulos independientes que tengan la posibilidad de comunicarse entre sí. Este tipo de descomposición condujo directamente a la programación orientada a objetos.

Con el tiempo ha ido surgiendo la necesidad de crear programas más grandes y complejos, lo que ha llevado a los desarrolladores a buscar una nueva forma de programar que les facilite crear sistemas a nivel empresarial y con complejas reglas de negocios. Para cubrir estas necesidades ya no era suficiente utilizar la programación estructurada, haciendo obsoleta la programación lineal. La Programación Orientada a Objetos (POO) surge de la evolución de la programación estructurada, la POO simplifica la programación basándose en dividir programas en pequeñas unidades lógicas de código a las que se les denominó objetos, que son independientes pero se comunican entre ellos por medio de mensajes.

Hoy en día la tecnología orientada a objetos ya no se aplica tanto a los lenguajes de programación como en el análisis y diseño igual que en la base de datos. Para lograr desarrollar un buen sistema, se tiene que aplicar esta tecnología en todo el proyecto, es por eso que es tan importante analizar y diseñar con la POO.

La programación orientada a objetos es una de las formas más populares de programar y ha sido muy bien aceptada por los desarrolladores de proyectos de software en los últimos años debido a las grandes capacidades y ventajas que ofrece.

1.4.2 Características de la Programación Orientada a Objetos.

En “POO”[4] todo problema aun aquellos sencillos de información, se resuelven como módulos de código gigante (clase) que contiene todo el código necesario como son variables, procedimientos, funciones, interfaces, etcétera para solucionar el problema.
La POO fomenta la reutilización y extensión del código, permite crear sistemas más complejos, relaciona el sistema al mundo real, facilita la creación de programas visuales, agiliza el desarrollo de software, facilita el trabajo en equipo, facilita el mantenimiento del software y proporciona conceptos y herramientas con las cuales se puede modelar y representar el mundo real tan fielmente como sea posible.

[4] Programación Orientada a Objetos. Flower Martin, Scout Kendall, UML Gota a Gota, Pearson Addison Wesley, México 1999, p. 8

1.4.3 Conceptos Básicos de la POO

Clase: Una clase es un prototipo, que define las variables y los métodos comunes a un cierto tipo de objetos. Las clases son las matrices de las que se pueden crear múltiples objetos del mismo tipo. La clase define las variables y los métodos comunes a los objetos de ese tipo, después, cada objeto deberá tener sus propios valores y compartirán las mismas funciones. Primero se debe crear una clase antes de poder crear objetos o ejemplares de esa clase.

Objeto: Objeto es una cosa tangible, que puede comprenderse intelectualmente y que tiene contenido al estado en sus variables.

Mensajes: Para poder crear una aplicación se necesita más de un objeto, estos objetos no pueden estar aislados unos de otros y para comunicarse esos objetos se envían mensajes. Los mensajes son cuando se mandan a llamar a los métodos del objeto con el se quiere comunicar para darle una orden.

Polimorfismo: El polimorfismo permite que se hagan muchas implementaciones de métodos según el tipo de objeto que se está indicando cuando se invoca al método correspondiente, permitiendo un solo mensaje enviado a distintos objetos y actúe de manera dependiente del objeto que recibe el mensaje.

Herencia: Es cuando una clase hereda las variables de su superclase y se le pueden agregar sus variables y métodos propios.

Método: Un método es aquél que establece las interfaces de comunicación con el código, es capaz de recibir parámetros de entrada y regresar algún tipo de dato. El método define el comportamiento de la clase.

1.4.4 Java.

En la actualidad Java es un lenguaje muy extendido de programación y cada vez es más importante tanto en la informática, como en el ámbito de Internet. Está desarrollado por la compañía Sun Microsystems y está enfocado a cubrir las necesidades tecnológicas más señaladas.

"Java es un lenguaje de programación que permite incorporar animación e interacción a las páginas WEB."[5]

Con Java los usuarios de Internet podrán acceder a programas de hoja de cálculo, tutoriales, animaciones o juegos interactivos.

Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas. Tiene la capacidad de soportar características de la orientación a objetos como encapsulamiento, polimorfismo y herencia.

[5] Carballar A. José, Internet. Libro del Navegante. RA-MA Editorial. Madrid 2000, p. 147

Es posible programar casi cualquier cosa con el lenguaje Java, ya que una de las características principales de Java es que es un lenguaje independiente de la plataforma, es decir, un programa en Java funciona en cualquier tipo de computadora, ya que se ha creado una máquina de Java que funciona como un puente entre el sistema operativo y el programa que posibilita el entendimiento perfecto. Java es capaz de funcionar en cualquier sistema operativo, es por eso que es tan importante par Internet, ya que los usuarios realizan una conexión desde diferentes máquinas con diferentes sistemas operativos.

1.4.5 Servlets.

Los Servlets son módulos que extienden los servidores que están orientados al servicio de petición/respuesta y podrían tener la responsabilidad de tomar datos de una forma de captura html y aplicarla a la lógica de negocios para actualizar una base de datos. Tienen la posibilidad de incrustarse en distintos tipos de servidores.

Java fue diseñado para trabajar sobre todas las plataformas, permitiendo que las aplicaciones sean movidas fácilmente de un sistema operativo a otro.

Los Servlets son una tecnología que puede ser alternativa a la programación CGI, ya que proporcionan una manera fácil de generar documentos dinámicos y tienen una mayor velocidad.

Los Servlets permiten la colaboración en grupos apoyando a los sistemas como las conferencias en línea.

También pueden reenviar peticiones a otros Servlets y a otros servidores, de este modo, los Servlets pueden particionar un servicio lógico en varios. Balancear la carga de trabajo y repartirla entre varios servidores que son imágenes del mismo contenido.

Los Servlets no están en riesgo de correr comando de shell no planeados. Los lenguajes compilados como Java proveen mejor seguridad que los lenguajes que interpretan scripts. Debido a que los Servlets son archivos de clases compilados dependiendo quien tenga acceso al servidor Web, se puede elegir entre instalar o no el código fuente.

Arquitectura de Servlet: Todos los Servlet implementan la interfaz Servlet directamente o extendiendo una clase que la implemente. Cuando un Servlet acepta una petición de un cliente, el Servlet recibe dos objetos: ServletRrequest que declara métodos que controlan al Servlet y a su comunicación de los clientes con el servidor y un objeto ServletResponse que encapsula la posterior comunicación del servidor con el cliente.

La interfaz ServletRequest permite al Servlet el acceso a información como los nombres de los parámetros usados por el cliente en el protocolo, así como los nombres del anfitrión remoto que hizo la petición y el servidor que la recibió.

La interfaz ServletResponse proporciona métodos al servlet para responder al cliente. Las interfaces que extienden a la interfaz ServletResponse le dan al servlet capacidades específicas del protocolo. Contiene métodos que permiten al servlet manipular información específica.

1.4.6 JAVAServer Page

JavaServer Page (JSP) es una tecnología desarrollada por Sun Microsystems basada en Scripts que utilizan una variante de Java y es diseñada para generar páginas web, de forma dinámica en el servidor y permite al código Java ser absorbido en el contenido estático. En JSP se escribe el texto que será devuelto en la salida incluyendo el código Java dentro de él, para que se pueda modificar o generar el contenido dinámicamente. Hay que tomar en cuenta que el código Java se incluye dentro de las marcas de etiqueta <% y %>.

JSP permite integrarse con clases Java, con esto separa en niveles las aplicaciones web, y almacena en las clases Java las partes que pudieran consumir más cantidad de recursos, así como las que requieren más seguridad.

Java es un lenguaje que puede ejecutarse en cualquier sistema, lo que le da mucha adaptabilidad a JSP. Sin embargo, JSP no se puede considerar un script al 100%, ya que antes de que se ejecute el servidor web compila el script y genera un servlet.

JSP, así como Java, está teniendo mucho peso en el desarrollo web profesional.

Siendo Microsoft la más directa competencia de Sun, ha visto en esta estrategia una amenaza, lo que le ha llevado a que su plataforma .NET incluya su lenguaje de scripts ASP.NET que permite ser integrado con clases .NET hechas en C++, VisualBasic o C# de la misma manera que JSP se integra con clases Java.

2. Ingeniería de software.

En este capítulo se hablará un poco sobre la Ingeniería del Software, su definición, sus orígenes y sus objetivos, así como algunas herramientas para el desarrollo de Software.

2.1 Definición de Ingeniería de Software

"La Ingeniería del Software es el proceso de construir aplicaciones de tamaño o alcance prácticos, en las que predomina el esfuerzo del software y que satisfacen los requerimientos de funcionalidad y desempeño."[6]

La Ingeniería del Software tiene el mismo conjunto de responsabilidades sociales que todas las otras ingenierías.

2.1.1 Orígenes de la Ingeniería del Software

La Ingeniería del Software se utilizó al final de la década de los 50's e inicios de los 60's.

Surge como disciplina en la década de los 70's cuando se hacen más graves los problemas de mantenimiento del software y de producción del mismo. A esta situación se le conoce como crisis del software.

El hecho de que no había documentación en los proyectos y las malas técnicas de programación crearon la crisis, llegaron a la determinación de afrontar el problema del software de modo sistemático, el cual corresponde a una modificación de un modelo que se había desarrollado durante los años 30 y que en el contorno del software conocido como ciclo de vida tradicional, que tenía como objetivo solucionar algún problema.

La idea de la etapa de análisis del problema es tener, desde distintas perspectivas, una visión más clara de los problemas y de los recursos con que se cuenta para solucionarlo.

El Diseño de la Solución, se orienta a evaluar distintas posibles soluciones del problema y subproblemas que se consideren en su análisis para llegar a la solución y pasar a la siguiente etapa que es la de implementación.

La etapa de implementación concreta la solución diseñada en el paso anterior.
Este modelo, que permite, en etapas tempranas de cualquier proyecto poder estimar los costos de recursos que considerará y la distribución de aquellos a lo largo del ciclo de vida; es acogido en los primeros años de la Ingeniería de Software y adaptado a las condiciones particulares que presenta el software como producto.

[6] J. Braude, Erick, Ingeniería del Software. Una perspectiva Orientada a Objetos. Ed. Alfa Omega, 2003, p. 17

En la figura 2.1 se muestra el orden de las etapas del ciclo de vida tradicional.

Figura 2.1. Ciclo de vida tradicional

2.1.2 Objetivos de la Ingeniería del Software

La ingeniería del software tiene como objetivos lograr una mejora en la calidad de los productos del software, definir una disciplina para garantizar el buen mantenimiento y la buena producción del software, que sea desarrollado dentro de los costos estimados y en el tiempo que se fijó en un inicio, proporcionar a los desarrolladores las bases para que logren la construcción de software de alta calidad de una manera eficiente, optimizar la productividad y el trabajo de los ingenieros del software y lograr un fácil control del proceso de desarrollo de software.

2.1.3 Optimización de la calidad de los productos de Software.

Normalmente las grandes aplicaciones tienen algunos defectos, ya que con el tiempo se van desgastando las aplicaciones físicas y es necesario especificar la calidad del software.

Por lo regular, un diseño de calidad se extiende, es de fácil manejo para lograr proporcionar funcionalidad adicional, evoluciona, con el fin de adaptarse fácilmente a los diferentes requerimientos, es movible para poder aplicarse en varios entornos y por último, es general y se puede aplicar a varias situaciones distintas.

En un código, la función de la calidad consiste en satisfacer los requerimientos establecidos, tener una tasa de defectos confiable y conocida, debe tener una reacción predecible a entradas ilegales, debe estar inspeccionado de manera íntegra por los ingenieros que no hayan participado en el desarrollo del sistema y debe ser aprobado de manera exhaustiva en formas independientes.

El objetivo es especificar estándares de aceptación y la creación de productos de software que satisfagan dichas especificaciones. Para lograrlo, se debe saber de qué manera cuantificar la calidad, cómo especificar metas en términos de estas cantidades, y de qué modo controlar el avance hacia las metas.

Una de las partes primordiales de la ingeniería es la cuantificación. En la ingeniería del software se utilizan las métricas para cuantificarlas, como los números de defectos fijos por mes, el número de funciones por clase, las líneas de código, etcétera.

Debido a la variación de la cobertura entre las métricas, suelen recolectarse varios tipos diferentes. Las métricas que casi siempre se incluirán son la cantidad de trabajo que se ha realizado, este debe ser medido físicamente como por ejemplo, las líneas de código, el tiempo que toma realizar dicho trabajo y la tasa de defectos, como por ejemplo, los defectos por mil líneas de código o defectos por página de documentos, etcétera.

Las clasificaciones que se deben incluir acerca del trabajo será en una escala de cero a diez.

Antes de realizar el esfuerzo, se pronostican los valores previstos o deseados par las métricas, después se comparan los resultados con los valores que se pronosticaron.

Procesos de aseguramiento de calidad:

Muchas organizaciones identifican un proceso separado de revisión exhaustivo y sistemático conocido como aseguramiento de calidad QA (Quality Assurance), además del trabajo que tiene cada ingeniero en cuanto al desarrollo y revisión del producto de software. El aseguramiento de calidad incluye revisiones, inspecciones y pruebas.

En la figura 2.2 se indica que el insumo de QA debe ser buscado desde el inicio de cada proyecto.

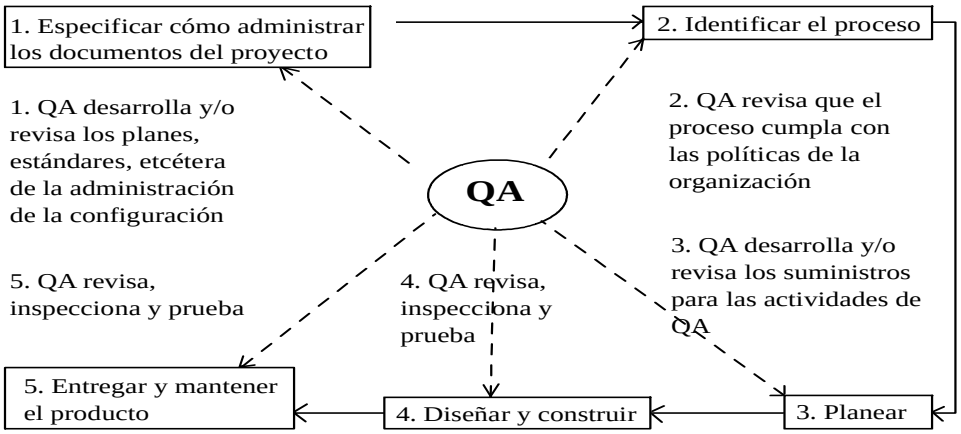


Figura 2.2. Intervención de Quality Assurance

El QA debe estar involucrado y garantizar que el proceso que se está utilizando es sólido y la documentación está actualizada. Uno de los representantes de QA puede participar frecuentemente en la inspección.

Una persona que no esté involucrada en el proyecto es quien debe realizar el QA.

a) Técnicas de la caja negra y la caja blanca

Las técnicas de caja negra de QA son las encargadas de verificar si el software cumple o no con sus requerimientos.

Las técnicas de caja blanca o caja de vidrio de QA son aplicadas en los componentes que forman la unidad que se está probando. Estas técnicas requieren que los ingenieros piensen en la estructura, la forma y el propósito del producto que analiza. Para esto, se deben usar métodos formales de inspección.

Estos conceptos se aplican a distintas actividades de aseguramiento de calidad, no sólo a las pruebas.

b) Inspección

La inspección es una técnica de caja blanca utilizada para el aseguramiento de calidad y consiste en examinar todas las partes del proyecto como pueden ser los requerimientos, los diseños, el código, etcétera, y así encontrar defectos o fallas.

Se deben llevar a cabo las inspecciones cuando producen la primera documentación del proyecto y cuando se haya introducido para mejorar el código, a esto se le llama inspección de código.

Existen 4 reglas que forman el principio de inspección.

1) *Detección de defectos.* Las inspecciones excluyen la reparación de defectos. El proceso de reparación debe ser responsabilidad del autor, no se le debe dedicar tiempo de inspección. Estas deben hacerse fuera del tiempo de inspección.

2) *Proceso de colegas.* Debe realizarse por un grupo de ingenieros. El producto debe someterse a una inspección y el resultado debe proyectar el mejor esfuerzo del ingeniero que realizó el producto, por lo que no debe ser un borrador o un diseño preliminar.

3) *Roles especificados.* Cada participante desempeña uno de los siguientes papeles:

   Moderador. Es responsable de verificar que se lleve a cabo la inspección de modo apropiado.

   Autor. Es responsable del trabajo en sí y de reparar los defectos encontrados.

   Lector. Debe conducir al equipo a través del trabajo en una forma adecuada e integral.

   Registrador. Es responsable de escribir las descripciones de los defectos y clasificarlos como el equipo lo decida.

4) *Preparación completa.* Las personas que participen en las inspecciones deben tener el mismo nivel de preparación que el autor. Las inspecciones deben trabajar al mismo nivel de detalle que el autor.

Los procesos de inspección comienzan con la planeación, las métricas de inspección que se recolectarán e identificar las herramientas que se usarán para registrar y analizar estos datos.

La preparación es donde los inspectores revisan el trabajo detalladamente e introducen los defectos encontrados en una base de datos que debe estar accesible desde la red junto con las descripciones y clasificaciones.

Una vez que los participantes están preparados, se lleva a cabo la junta de inspección.

Si en la junta se define que las fallas son tan grandes, requerirán una nueva inspección y se reciclará a través del proceso.

En la junta final el moderador y el autor confirman que los defectos han sido reparados.

c) Estándar de IEEE para planes de aseguramiento de la calidad del software

El uso de los estándares de calidad existentes se debe a que son muy útiles las listas de verificación de cosas que hacer y buscar, para asegurar que se han cubierto las bases de calidad.

El estándar IEEE 730-1989 se refiere al plan de aseguramiento de la calidad del software. Este estándar está dirigido a grandes proyectos, pero puede ser utilizado en los pequeños, y esto ayudará a recordar los factores que se deben incluir.

Este estándar especifica quién va a ser el responsable de la calidad, podrá ser un gerente, un grupo o una organización; también indicará la documentación que se requiere, la técnica que se utilizará para asegurar la calidad, las inspecciones y la demostración de lo que es correcto, las pruebas, los procedimientos que se deberán seguir para la administración del proyecto, las reuniones, auditorias, revisiones, etcétera.

2.2 Componentes de la Ingeniería del Software

Los componentes de software son construidos mediante lenguajes de programación con una gramática definida, reglas bien formadas de sintaxis, semántica y un vocabulario limitado.

Una de las tendencias predominantes en la Ingeniería del Software ha sido el surgimiento de nuevas herramientas y procesos, también el crecimiento de aplicaciones de todo tipo, incluyendo las que están asociadas con Internet. A pesar de que han llegado estas tendencias, permanecen siempre presentes las actividades básicas para cumplir el objetivo fundamental que es la construcción del producto de software de calidad.

2.2.1 Procesos de la Ingeniería del Software

“El proceso de la Ingeniería del Software es un conjunto de etapas, actividades, prácticas y métodos que deben guiar a los ingenieros y a sus herramientas de software en la producción de Software con la finalidad de llegar a un solo objetivo, la obtención de un producto de software, de calidad.”[7]

La política, economía, tecnología, costumbres, cultura, y la Ingeniería del software son factores que determinan un buen proceso del Software, que tiene beneficios como la mejora de los recursos humanos, reducción de la repetición del trabajo, proporciona un mayor tiempo para emplearlo en los problemas que requerirán mayor energía y creatividad, proporcionará un mecanismo para aprender de experiencias ajenas, además incrementará la probabilidad de introducir tecnología con gran éxito.

[7] E. Fairley, Richard, Ingeniería del Software, Ed. Mc Graw-Hill, 1987, p.46

Se deben tener en cuenta unos componentes para lograr el proceso de Ingeniería de Software, que son los formalismos o teorías para desarrollar los demás componentes, las técnicas y metodologías de desarrollo y mantenimiento, componentes de administración de proyecto y herramientas CASE.

La ingeniería del software es una colección de tecnologías entrelazadas que se deben considerar como un sistema.

Uno de los objetivos ha sido, por varios años, encontrar procesos o metodologías predecibles y repetibles que mejoren la productividad y la calidad.

La ingeniería de software requiere llevar a cabo muchas tareas, entre ellas están los siguientes pasos del proceso:

Análisis de requisitos: Se requiere tener gran habilidad y experiencia en la Ingeniería de Software para reconocer requisitos incompletos o contradictorios. De modo que en esta etapa se debe analizar y extraer los requerimientos para saber sobre qué se va a trabajar.

Especificación: Se trata de describir detalladamente el software a desarrollar en una forma rigurosa. En la realidad, la mayoría de las buenas especificaciones han sido escritas para entender y afinar aplicaciones que ya estaban desarrolladas.

Diseño y arquitectura: Consiste en incorporar consideraciones de la implementación tecnológica, como el hardware, la red, etc.

Programación: Se refiere a reducir un diseño a código, aunque, en teoría parece, no es necesariamente la porción más larga.

Prueba: Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo todo completo.

Documentación: Realización del manual de usuario y posiblemente técnico con el propósito de mantenimiento futuro y ampliaciones al sistema.

Mantenimiento: Esto puede llevar más tiempo incluso que el desarrollo inicial del software, ya que consiste en mantener el software en buen estado y mejorarlo si llega a surgir algún error y para cumplir con nuevos requerimientos. Una pequeña parte de este trabajo consiste en la corrección de errores La mayor parte consiste en extender el sistema para hacer nuevas cosas.

2.2.2 Modelos del proceso del Software

El desarrollo de software se divide en 4 etapas distintas y es como un bucle de resolución de problemas como se explica en la figura 2.3, donde el status quo representa el estado actual de sucesos.

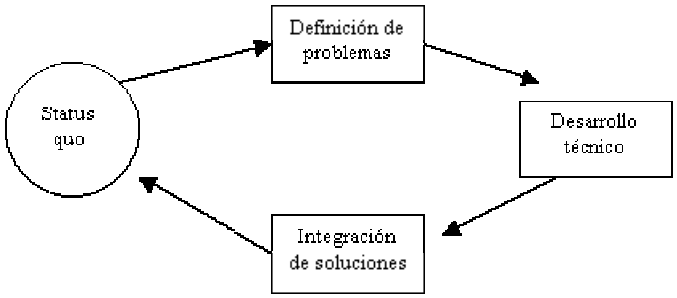


Figura 2.3. Bucle de solución de problemas.

Modelo en espiral: Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo y conforme se va avanzando en las iteraciones, las versiones de la ingeniería del software serán cada vez más completas. En la figura 2.4 se puede ver dicho modelo.

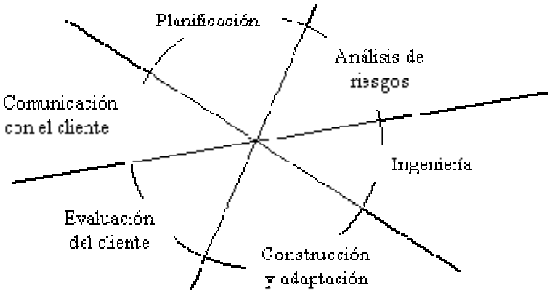


Figura 2.4. Modelo en espiral

Modelo lineal secuencial: En la figura 2.5 se muestra un diagrama con este modelo que también es denominado modelo de cascada.

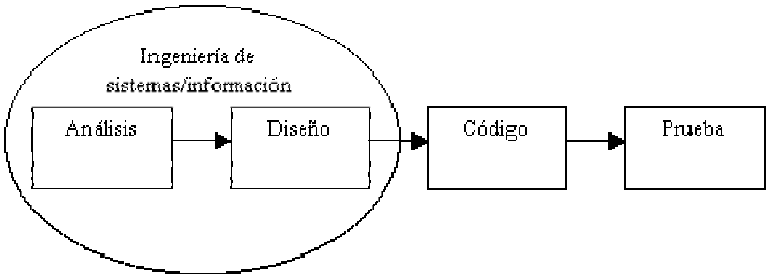


Figura 2.5. Modelo Secuencial

Modelos de proceso evolutivo del software: El software, al igual que todos los sistemas complejos, evoluciona con el tiempo. Los requisitos cambian con frecuencia; las estrictas fechas del mercado hacen que no sea posible finalizar un producto de calidad, por lo que se debe introducir una versión limitada para cumplir la presión competitiva, pero todavía se tienen que definir los detalles de extensiones del sistema. Debido a estas situaciones, los ingenieros del software necesitan un modelo de proceso diseñado para adaptarse a que un producto debe ir evolucionando con el paso del tiempo.

Modelo de construcción de prototipos: Ofrece su enfoque a través de las normas de construcción de modelos y se muestra en la figura 2.6.

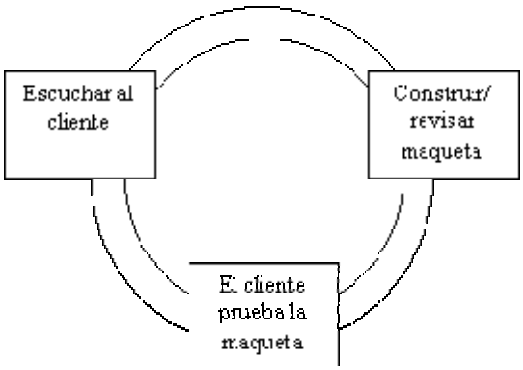


Figura 2.6. Modelo de construcción de prototipos

Modelo RAD (Rapad Application Development): El Desarrollo Rápido de Aplicaciones (DRA) es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando un enfoque de construcción del producto de software que está basado en componentes.

2.2.3 Procesos de desarrollo en la ingeniería de software

Un proceso de desarrollo de software es aquel en el que las necesidades de los usuarios se traducen en requerimientos del software y son transformados en diseño, este se implementa en código, el cual se prueba y certifica para su óptimo uso operativo. El proceso de desarrollo define quién está haciendo cada cosa, cuándo lo debe hacer y cómo puede alcanzar el objetivo.

El proceso de desarrollo de software requiere un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso se le llama el ciclo de vida del software y comprende como mínimo cuatro fases: La concepción, que define el alcance del proyecto y desarrolla un caso de negocio. La elaboración, que define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción que crea el producto y la transición, que transfiere el producto a los usuarios.

El objetivo de un proceso de desarrollo es aumentar la calidad del software en todas las fases por las que pasa a través de una mayor transparencia y control sobre el proceso. Es labor del proceso de desarrollo hacer que sean aplicadas dichas medidas y así subir la calidad.

La cantidad y la variedad en los procesos de desarrollo han ido en aumento constante. Con el paso del tiempo se desarrollaron dos corrientes en cuanto a los procesos de desarrollo se refiere, los métodos ligeros y los métodos pesados.

Los métodos pesados intentan llegar a un objetivo común mediante la documentación y el orden.

Los métodos ligeros o métodos ágiles, intentan mejorar la calidad del software mediante la comunicación directa entre las personas que intervienen durante el proceso.

2.2.4 Racional Unified Proccess

Siendo RUP un proceso pesado, es uno de los procesos más generales de los que actualmente existen, está pensado para adaptarse a cualquier tipo de proyecto.

Siguiendo la metodología RUP, un proyecto se divide en 4 etapas que son la intercepción, que es la puesta en marcha, la elaboración, que se refiere a la definición, el análisis y el diseño, la construcción que es la implementación y la transición que es el fin del proyecto y la puesta en producción.

En cada una de las fases se ejecuta una o varias iteraciones y dentro de cada iteración se sigue el modelo de cascada para los flujos de trabajo que necesitan las nuevas actividades.

El modelado del negocio, análisis de requisitos, análisis y diseño, implementación, test, distribución, gestión de configuración y cambios, gestión del proyecto y gestión del entorno son nueve actividades a realizar en cada una de las fases del proyecto.

Y el flujo de trabajo entre las actividades se basa en los diagramas de actividad.

El proceso define una serie de actividades distribuidas entre los miembros que integran el proyecto y que definen las tareas de cada uno y el resultado que se espera de ellos.

En la figura 2.7 se puede ver el flujo de trabajo de RUP.

Figura 2.7. Flujos de trabajo de RUP

RUP se basa en casos de uso para describir lo que se espera del software y se orienta a la arquitectura del sistema, documentándose lo mejor posible, basándose como herramienta principal en UML.

RUP es un proceso grande, antes de usarlo es necesario adaptarlo a las características de la empresa.

2.2.5 Extremme Programing

La metodología XP está recomendada para aumentar la velocidad de desarrollo de un producto que fundamenta su desarrollo en los casos de prueba y en las historias de usuario.

Lo más importante en el desarrollo de una aplicación, es el uso adecuado de una metodología que administre y garantice el cumplimiento de los requerimientos especificados al inicio del proyecto, por medio del uso de alguna tecnología apropiada.

XP intenta disminuir el riesgo de error del proceso por medio de la disposición permanente de un representante del cliente que esté a disposición del equipo de desarrollo y que esté en condiciones de contestar rápida y correctamente a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones.

Mientras que el RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

XP define *UserStories* como base del software a desarrollar. Estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del software, que pueden describir el modelo, dominio, etc. A partir de las UserStories se crea un plan de liberación o entrega del software entre el equipo de desarrollo y el cliente.

Para cada liberación de software se discutirán los objetivos con el representante del cliente y se definirán las iteraciones necesarias para cumplir con los objetivos. El resultado de cada iteración es un programa que se transmite al cliente para que lo evalúe. En base a su opinión se definen las siguientes iteraciones del proyecto, y si el cliente no está contento se adaptará nuevamente el plan de liberación de software e iteraciones hasta que el cliente de su aprobación y quede satisfecho con el resultado.

Junto a los UserStories están los escenarios de pruebas que describen el escenario contra el que se comprueba la realización de las UserStories. UserStories y casos de pruebas son la base sobre la que se asienta el trabajo del desarrollador.

En cada iteración se deberán escribir las pruebas de forma que se puedan ejecutar automáticamente y se compruebe la corrección del software antes de cada liberación de software. Esto es indispensable en XP, ya que la funcionalidad del software se escribe cuando las pruebas para su corrección estén listas.

Se puede analizar la vista general de Extreme Programming Project en la figura 2.8.

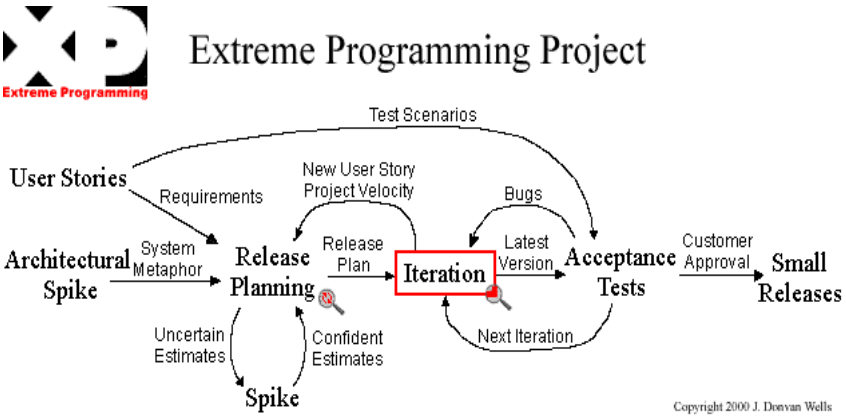


Figura 2.8. Vista general de XP

La codificación del software en XP se produce siempre en parejas de dos programadores por una computadora, por lo que se espera que la calidad del mismo suba en el mismo momento de escribirlo.

El código pertenece al equipo en completo, no a un programador o pareja, cada programador puede cambiar cualquier parte del código en cualquier momento si así lo requiere. Las parejas se rotan a lo largo del proyecto para que cada integrante del equipo aprenda a trabajar con todo el proyecto.

El objetivo ideal sería que cada integrante del equipo trabaje al menos una vez con cada uno de los demás integrantes y con cada componente software, para que todos conozcan por completo el sistema.

En XP se programará solo la funcionalidad que se requiera para la liberación del software actual. Se sigue un diseño evolutivo con la premisa de conseguir la funcionalidad deseada de la forma más sencilla posible. Este diseño evolutivo hace que se trabaje exclusivamente en función de las necesidades del momento.

2.3 Herramientas para el desarrollo.

En este tema hablaremos un poco de las herramientas de desarrollo como la tecnología CASE y una pequeña introducción de UML.

2.3.1 Tecnología CASE y entornos de desarrollo.

La tecnología CASE es un conjunto de herramientas de software que automatiza las tareas del desarrollo de software y es utilizada en cualquiera de las fases de desarrollo del sistema de información.

La tecnología CASE corresponde a la Ingeniería del Software asistido por computadora y proporcionan métodos automáticos que sirven para diseñar y documentar las técnicas de Programación Orientada a Objetos con el objetivo de proveer un lenguaje para describir un sistema complejo, que sea suficiente para generar todos los programas necesarios.

La idea es proporcionar un conjunto integrado de herramientas que enlazan y automatizan todas las fases del ciclo de vida del software y su administración.

Los objetivos de la tecnología CASE son aumentar la productividad de desarrollo, dar buena calidad a los productos desarrollados, reducir el coste del software, automatizar los chequeos de errores, el desarrollo de las aplicaciones, automatizar tareas de desarrollo, automatizar la generación de documentación y código, dar portabilidad al software, implantar metodologías de desarrollo, datos reutilizables y compartidos, administrar el proyecto y la Ingeniería hacia atrás.

Entre los tipos de herramientas CASE tenemos:

Herramientas Front-end: Son herramientas de planificación y definición de requisitos, herramientas de análisis y diseño, así como herramientas de creación de modelos y generación de prototipos.

Herramientas back-end: Compiladores, Generadores de estructura y código.

Gestión: Gestión de proyectos o ciclo de vida, Gestión de proyectos (dirección proyecto), Gestión de requisitos, Gestión de cambios y configuraciones.

Workbench: Es el conjunto de herramientas integradas que interactúan unas con otras de forma consistente conforme a un conjunto de estándares y ante el usuario conforman un bloque único sin funciones o mensajes duplicados. En lo que refiere a las capacidades funcionales básicas el workbench proporciona asistencia mediante computadora para el desarrollo, mantenimiento y administración de sistemas software de forma integrada y debe tener las siguientes características: Interfaz gráfico para dibujar diagramas estructurados, una enciclopedia para almacenar y administrar toda la información del sistema software, un conjunto integrado de herramientas que comparten una interfaz de usuario común, herramientas para asistir en cada fase del ciclo de vida, herramientas para prototipo, herramientas para administración del proyecto, generación automática de código desde las especificaciones de diseño, soporte, metodología, ciclo de vida del software con verificación incorporada en la herramienta.

La enciclopedia es un elemento de mecanismo clave para almacenar la información referente a un sistema software e incluye información referente a planificación estratégica, diseño, administración del proyecto, análisis e implementación, y es un punto clave para tener una productividad alta y permitir obtener información para los desarrolladores en caso de que se necesite. Es el único lugar donde se almacena toda la información del sistema, de forma disponible para quien la necesite.

Las funciones que realiza la biblioteca son: almacenamiento, acceso, actualización, análisis e informes sobre la información del sistema, administra la información del sistema y mantiene toda la información necesaria para crear, modificar y mantener un sistema software, e incluye información relativa a problemas a resolver, sistema de solución emergente, contexto organizativo, historia y recursos del proyecto y proceso software a seguir.

Si las herramientas no soportan una metodología son de poca utilidad. La tecnología de trabajo CASE constituye un ambiente completo que incluye hardware y software. Su objetivo principal es incrementar la productividad y mejorar la calidad.

2.3.2 UML

"UML"[8] es un lenguaje de modelado. El lenguaje de modelado es la notación de que se valen los métodos para expresar los diseños. El proceso es la orientación que nos dan los pasos a seguir para realizar el diseño.

[8] Unified Modeling Language o Lenguaje Unificado de Modelado. Fowler, Martin y Scout, Kendall, UML Gota a Gota, Addison Wesley Longman de México, 1999, p. 1

UML es un lenguaje que permite:

Especificar (construir un modelo de forma precisa, no ambiguo y completo.)

Visualizar (lenguaje gráfico que facilita la comunicación)

Construir (permite la simulación del sistema y la generación automática de código en algún lenguaje de programación).
Documentar: UML necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, y debe ser tan simple como sea posible.

3 Lenguaje unificado para modelado de Software (UML)

En el capítulo anterior se dio una pequeña introducción de lo que es UML, en este capítulo se habla un poco sobre el surgimiento y algunas características de UML, así como algunos tipos de diagramas por los que está conformado.

3.1 Surgimiento de UML

Este Lenguaje Unificado de Modelado o UML surgió a finales de la década de 1980 y principios de 1990.

UML unifica principalmente los métodos de Grady Booch, Jim Rumbaugh (OMT) e Ivar Jacobson (“Los tres amigos”), su alcance llegará a ser más amplio.
Actualmente el UML está en proceso de estandarización con el OMG (Object Management Group o grupo de la administración de objetos).

Con muchos desarrollos en el software, los objetos estuvieron guiados por los lenguajes de programación; esto causaba un problema para adaptar los métodos de diseño a un lenguaje orientado a objetos, y consideraban importantes las técnicas para lograr un buen análisis y diseño de los productos de software.

Algunos de los libros más sobresalientes acerca del análisis orientado a objetos y los métodos de diseño aparecieron entre 1988 y 1992, y son los siguientes:

Rally Shlaer y Steve Mellor (1989 y 1991) escribieron un par de libros sobre análisis y diseño; la evolución del material de estos libros ha dado como resultado su enfoque de diseño recursivo (1997).

Meter Coad y Ed Yourdon escribieron libros en los que desarrollaron el enfoque hacia los métodos ligeros orientados a prototiopos de Coad.

La comunidad Smalltalk de Portland, Oregon, aportó el diseño guiado por la responsabilidad (Responsability-Driven Design) y las tarjetas de clase-responsabilidad-colaboración (Class-Responsability-Colaboration) (CRC) (Beck y Cunningham 1989).

Grady Booch. En sus libros se daban varios ejemplos sobre el desarrollo de sistemas en Ada.

Jim Rumbaugh escribió un libro sobre un método llamado técnica de modelado de objetos (object Modeling Technique) (OMT).

Los libros de Jim Odell, escritos junto con James Martin, se basan en su amplia experiencia en los sistemas de información de negocios y de ingeniería de información. El resultado fue el libro más conceptual de todos.

Ivar Jacobson escribió con base en la experiencia adquirida en conmutadores telefónicos para Ericsson e introdujo en el primero de sus libros el concepto de casos de uso (use cases).

Todos estos modelos eran muy similares, pero aún así existían algunas diferencias en cuanto a que los mismos conceptos básicos aparecían con diferentes denominaciones.

Grady y Jim declararon que iban a lograr la estandarización a la manera de Microsoft. Prepararon la primera descripción pública de su método integrado: la versión 0.8 de la documentación del Método unificado (Unified Method); anunciaron que racional Software había comprado Objectory y que Ivar Jacobson se uniría al equipo unificado.

Durante 1996, Grady, Jim e Ivar, construyeron su método y le pusieron el nombre Unified Modeling Language (UML), lenguaje unificado de modelado.

En 1997, varias organizaciones entregaron sus propuestas de estandarización de métodos, con el fin de simplificar el intercambio de modelos. Como su propuesta al OMG, la Racional liberó la versión 10.0 de la documentación del UML.

3.2 Características de UML

"El UML es un lenguaje Unificado de Construcción de modelos que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional."[9] UML facilita la comunicación entre el cliente y el diseñador para expresar mejor sus requerimientos y poder señalar cambios si es que el analista no ha captado al 100% sus ideas.

UML facilita la organización del proceso de diseño con mayor facilidad con el fin de que sea comprendido por los clientes, analistas y desarrolladores; se conforma por elementos gráficos que se combinan para la creación de diagramas, para esto cuenta con varias reglas y finalmente se describe gráficamente lo que hará el sistema, pero no describe la manera en que se implementará.

Para crear una aplicación de software se necesita describir el problema y las necesidades o requerimientos. El análisis se centra en una investigación del problema, no en la manera de definir una solución. Para desarrollar una aplicación es necesario hacer descripciones detalladas y de alto nivel de la solución lógica y saber de qué manera satisface los requerimientos y las restricciones; es una solución lógica de cómo el sistema cumple con los requerimientos.

Para el desarrollo de software orientado a objetos es necesario realizar un análisis y diseño orientado a objetos.

[9] Schmuller, Joseph, Aprendiendo UML en 24 horas, Pearson Educación, México 2000, p. 29

UML: este tiene como finalidad modelar cualquier tipo de sistemas mediante los conceptos de la orientación a objetos y debe ser entendible para los humanos y máquinas. Consta de todos los elementos y diagramas que sirven para modelar los sistemas en base a las teorías de la orientación a objetos.

El modelado UML es flexible al cambio y permite crear componentes plenamente reutilizables. Si se construyen correctamente los modelos orientados a objetos, deben ser fáciles de comunicar, verificar, validar, expandir y cambiar.

Por citar a algunos de los diagramas por los que está conformado UML tenemos los diagramas de casos de uso, diagramas de secuencia, diagramas de clases, diagramas de colaboración, entre otros.

3.2.1 Diagrama de Casos de Uso

Su principal función es capturar los requerimientos del sistema, describir la funcionalidad y los actores que intervienen en él. Describe la secuencia de eventos de un actor que utiliza un sistema para completar un proceso, representa gráficamente la funcionalidad del sistema como la vería el usuario, y la manera en que el cliente trabajará con el sistema. Los diagramas de casos de uso están expresados desde el punto de vista del actor y en la figura 3.1 se muestran los elementos con los que consta dicho diagrama.

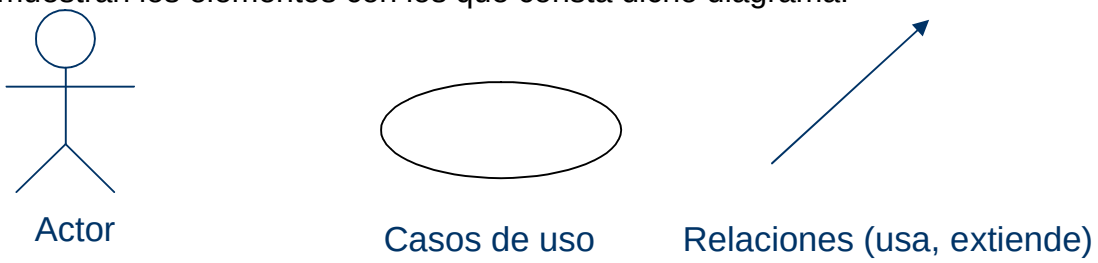


Figura 3.1 Elementos que expresan los casos de uso

Donde el Actor es el agente externo, el usuario, no necesariamente representa a una persona en particular, representa quién realiza las operaciones identificadas.

Casos de Uso: Es una tarea específica que se realiza tras una orden de algún agente externo como un actor u otro caso de uso.

Relación extend: Es utilizada cuando se tiene un caso similar a otro, pero más especializado.

Relación usa: Es usada cuando se tiene un comportamiento similar en más de un caso de uso y no se quiere copiar la descripción de tal conducta.

La realización de un diagrama de casos de uso se basa en el eunciado general del problema, la entrevista no dirigida, las entrevistas dirigidas y el análisis de la tarea

Puede existir alguna jerarquía entre actores, donde un actor puede ejecutar todos los casos que ejecute otro actor.

3.2.2 Diagramas de Secuencia.

En un diagrama de secuencia, un objeto se muestra como caja en la parte superior de una línea vertical punteada. Esta línea vertical se llama línea de vida del objeto, que representa la vida del objeto durante la interacción. Esta forma fue popularizada por Jacobson.

Cada mensaje representa mediante una flecha entre las líneas de vida de dos objetos. El orden en el que se dan estos mensajes transcurre de arriba hacia abajo. Cada mensaje es etiquetado por lo menos con el nombre del mensaje; pueden incluirse también los argumentos y alguna información de control.

3.2.3 Diagrama de Colaboración

En estos diagramas, los objetos ejemplo son mostrados como iconos. Consta de unas flechas que indican los mensajes enviados dentro del caso de uso dado. La secuencia se indica enumerando los mensajes.

El numerar los mensajes dificulta más ver la secuencia que poner las líneas verticales en la página. La disposición espacial del diagrama permite mostrar otras cosas mejor. Se puede mostrar cómo se vinculan los objetos entre sí y emplear la disposición para sobreponer paquetes u otra información.

3.2.4 Diagrama de Clases

Los diagramas de clase constan de métodos y atributos, facilitan la representación a partir de la cual el desarrollador podrá trabajar, además, posibilita que los clientes indiquen detalles importantes a los problemas que desean que se resuelvan. Estos diagramas son de uso extendido y están sujetos a una amplia variedad de conceptos de modelado. Describe los objetos que existen en el sistema y las distintas clases de relaciones estáticas que hay entre ellos. Muestran las operaciones y los atributos de las clases y las restricciones a que están sujetas de acuerdo a la forma en que se relacionan con los objetos y hacen una descripción grafica de las características de las clases de software en una aplicación.

En los diagramas de clase el Nombre de Clases Inicia con mayúscula; si el nombre tiene más de una palabra, se unen, un ejemplo lo tenemos en la figura 3.2:

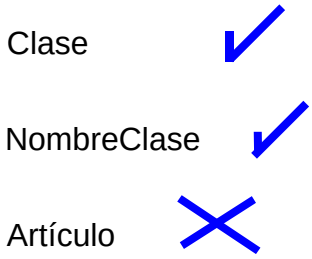


Figura 3.2 Manera correcta para nombrar una clase

Atributos: Dependiendo de la comunicación y visibilidad entre ellos y el medio que los rodea, la manera en la que están divididos se muestra en la tabla 3.1:

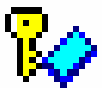
Tipos de Atributo	**Descripción:**	**Símbolo**
Públicos (+)	Son accesibles tanto dentro como fuera de la clase.	
Protegidos (#)	No se puede acceder desde fuera de la clase, pero si por métodos de la clase y subclases.	
Privados (-)	Sólo es accesible desde dentro de la clase (sólo sus métodos los podrán acceder).	

Tabla 3.1 Tipos de atributos

Métodos: Los Métodos son la manera en que interactúa la clase con el medio que lo rodea.

Tipos de Métodos:	**Descripción:**	**Símbolo**
Públicos (+)	Indica que el método será visible, tanto dentro como fuera de la clase.	
Protegidos (#)	Se puede acceder por métodos de la clase, así como métodos de las subclases.	
Privados (-)	Sólo es accesible desde dentro de la clase.	

Tabla 3.2 Tipos de métodos

Un ejemplo de las clases de diseño se muestra en la figura 3.3:

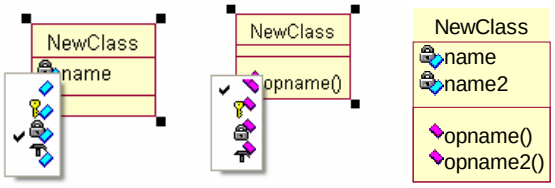


Figura 3.3 Ejemplo de Clases de Diseño

Estos son los estereotipos de las clases de diseño:

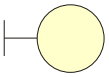
Clase	Estereotipo
Boundary (Botones)	
Control (Métodos)	
Entity (Atributos y Métodos)	

Tabla 3.3 Clases de diseño

3.2.5 Diagrama de Estados.

Son conocidos como motor de estados, describe los diversos estados en los que puede encontrarse un cierto objeto. Este diagrama junto con el de secuencia representa información estática.

El diagrama de estados es una técnica que describe el comportamiento de un sistema y todos los estados posibles en los que puede entrar un objeto en particular así como la manera en que cambia el estado del objeto como resultado de los eventos que llegan a él.

En un diagrama de estados parte de un estado (pantalla) y según el evento o acción del usuario se pasa al siguiente. Los eventos son los métodos que tiene cada pantalla.

El diagrama de estados permite a los analistas, diseñadores y desarrolladores la comprensión más sencilla en el comportamiento de los objetos del sistema.

Los desarrolladores, en particular, deben saber la manera en que los objetos se comportarán. Aseguran que no se tenga que adivinar lo que se supone harán los objetos.

Los elementos que integran un diagrama de estados se muestran en la figura 3.4:

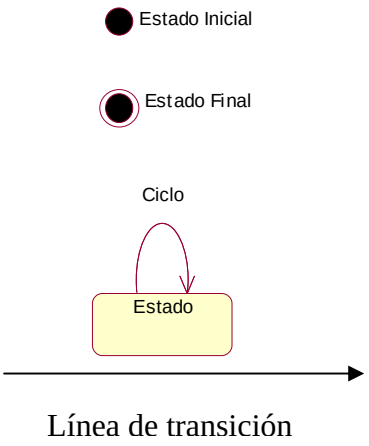


Figura 3.4 Elementos que integran un diagrama de estados

3.2.6 Diagramas de Actividades

El diagrama de actividades permite seleccionar el orden en que se van a hacer las cosas. Indica las reglas esenciales de secuenciación que hay que seguir. Es una parte integral del análisis del sistema. Este diagrama proviene de los diagramas de flujo, en él, se plasman las actividades que ocurren dentro de un caso de uso o dentro del comportamiento de un objeto y se muestra la secuencia de pasos, procesos, puntos de decisión y bifurcaciones.

Los elementos que lo integran son expresados en la figura 3.5:

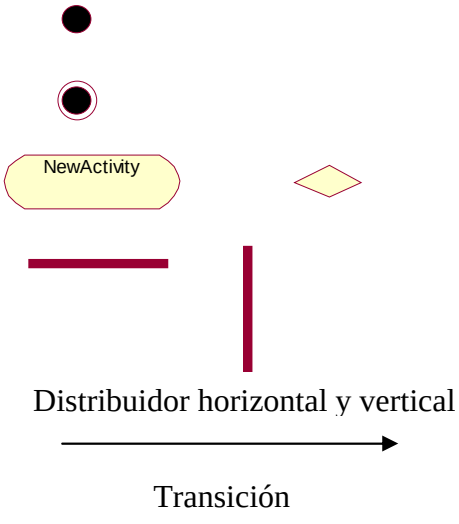


Figura 3.5 Elementos que integran un diagrama de Actividades

En las decisiones mutuamente excluyentes la condición va dentro de corchetes junto a la ruta correspondiente.

Las Rutas concurrentes separan actividades que se ejecutan al mismo tiempo y que luego se unen. Para representar esta división se usa una línea gruesa perpendicular y las rutas partirán de ella. Para representar la unión, ambas rutas apuntarán a otra línea gruesa paralela a la anterior.

4. Análisis del Sistema

En este capítulo se hace un análisis general del proyecto a realizar, en el cual se agregan algunos diagramas que considero de importancia para analizar el sistema.

4.1 Enunciado General

Sistema para control y realización de órdenes de empaque de los productos terminados en el área de Planeación.

4.2 Problemática Actual

En una compañía que empaca productos químicos, actualmente el sistema que se utiliza para llevar a cabo las órdenes de empaque de los productos es muy deficiente por las siguientes razones:

Las razones principales son por la manera de llevar a cabo una orden, ya que es de forma manual y en caso de existir algún error tiene que repetirse para evitar confusiones y sólo se registra la orden en un archivo de Excel al cual sólo tiene acceso el departamento de planeación, además de que la orden no proporciona los datos suficientes, por ejemplo, los materiales que se van a utilizar y los trabajadores no pueden efectuar el empaque correctamente, además la orden no indica la prioridad que tiene el producto, lo cual es una pérdida de tiempo, pues pueden empacar productos que tienen fecha de entrega más tarde que otros que son más urgentes.

Otra de las fallas es que la orden no indica para qué cliente es el producto empacado y esto implica tener que separarlo hasta que se envió al almacén de producto terminado pudiendo ahorrar tiempo al enviarlo separado desde el momento de su empaque.

4.3 Requerimientos Funcionales.

Este sistema va a llevar un control de las órdenes emitidas desde el área de Planeación hacia las diferentes áreas que participan en el proceso.

En primer lugar existirá una base de datos con los productos que se empacan y de los materiales indicando el número de catálogo, descripción y unidad de medida así como de los clientes de la compañía.

Para llevar a cabo el empaque de los productos, en la orden se indicará lo siguiente:

La fecha de creación de la orden y la fecha de entrega, el catálogo, cantidad en piezas y descripción del producto, la unidad de medida en que se va a empacar y los materiales a utilizar, el idioma en que se elaborarán las etiquetas y el cliente que requiere el producto.

La orden no se imprimirá, ya que las personas involucradas podrán tener acceso al sistema, para ello tendrán un perfil de usuario que cubrirá sus necesidades con sus respectivo usuario y password para controlar la seguridad y evitar que tengan acceso a la base de datos o a menús no necesarios para ellos.

4.4 Requerimientos no Funcionales.

Este sistema se elaboró y diseñó aplicando los conocimientos que se adquirieron durante el Diplomado, utilizando las siguientes herramientas:

Para diseñar y documentar el sistema se utilizó UML
Patrón de arquitectura de software MVC (Modelo Vista Controlador), el cual consiste en separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes:

Modelo: Contiene las reglas de negocio y notifica a las vistas el cambio de información si es que existe.

Vista: Se encarga de manejar la representación visual de los datos representados por el modelo y muestra los datos al usuario.

Controlador: Responde a eventos, a acciones del usuario, invoca cambios en el modelo y probablemente en la vista.

Este patrón de arquitectura se diseñó para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados en los mismos datos, el modelo, las vistas y los controladores son tratados como entidades separadas, de este modo, cualquier cambio que se produzca en el modelo se refleja automáticamente en cada una de las vistas.

Para el desarrollo del sistema se utilizó el programa compilador NetBeans 5.0, el cual trae consigo el kit de desarrollo de Java.

La base de datos la realicé en Mysql y el lenguaje de programación que utilicé fue Java.

4.5 Propuesta de Solución

El hecho de que el sistema nos indique la información completa optimizaría el tiempo de la empresa en general, ya que se evitaría el papeleo impreso y el estar buscando la información en archivos físicos, bastaría con consultar la información en el sistema.

Yo considero que el sistema sería muy factible si se alimenta correctamente por los diferentes usuarios, ya que cada uno será responsable de darle el uso adecuado con el fin de que el sistema completo proporcione la información correcta y necesaria y se cumpla el objetivo principal que es el empaque del producto terminado en el menor tiempo y con la mayor eficiencia posible.

4.6 Diagrama general de Casos de Uso

En este tema se agrega el diagrama General de Casos de uso, el cual se muestra en la figura 4.1, donde se puede ver que consta de 4 actores y 14 casos de uso.

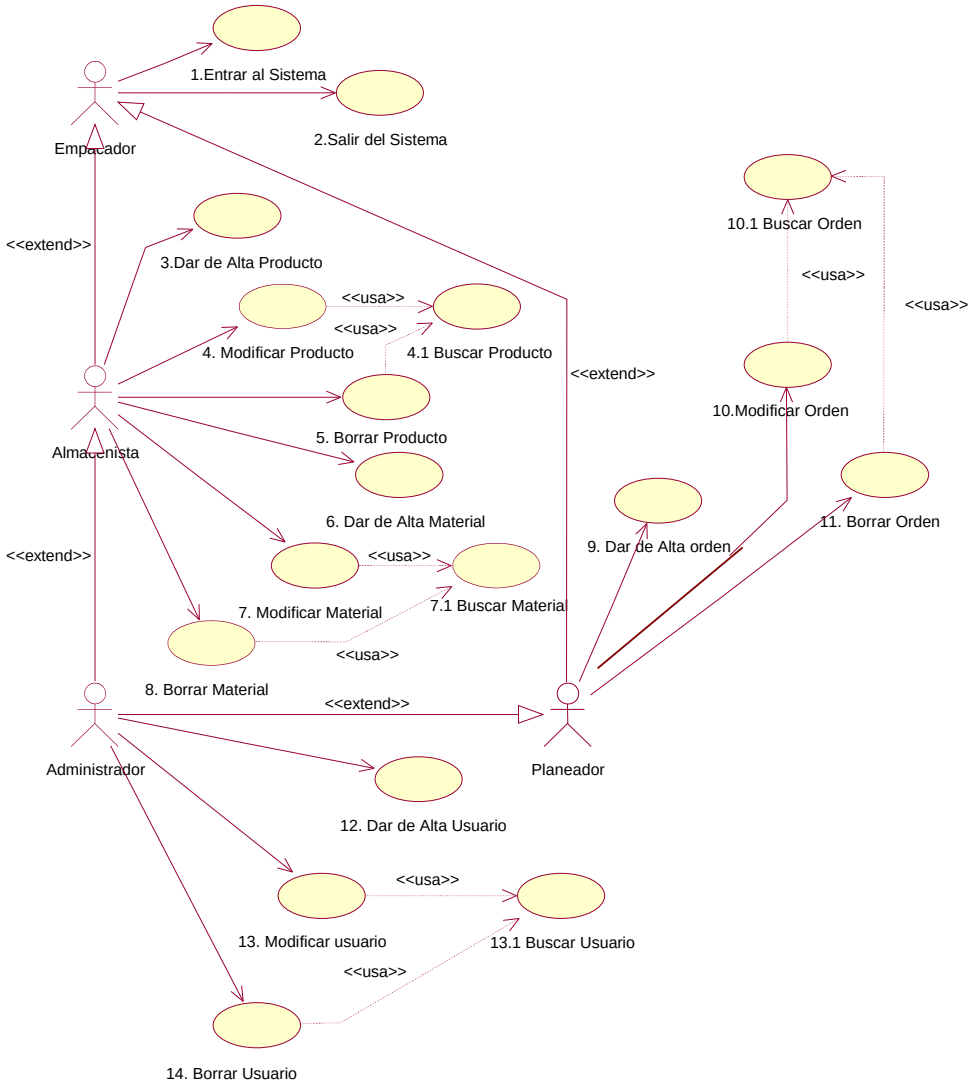


Figura 4.1 Diagrama General de Casos de Uso

4.6.1 Casos de uso formato expandido.

Aquí se muestran los casos de uso formato expandido del sistema, el cual consta de 14 casos.

Caso de uso: **1. Entrar al Sistema**

Actor: Empacador

Descripción: El empacador debe escribir su usuario y password, para Ingresar al sistema.

Precondiciones:

- El empacador debe contar con un usuario y password válido.
- El empacador debe entrar al sistema.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Entrar a la página inicial del sistema.	2	Mostrar la página de acceso, donde se encuentra el formulario de usuario y password.	
3	Captura el usuario y password de acceso. Acepta los datos.	4	Despliega la página principal del sistema.	E1, E2

Excepciones:

Id	Nombre	Acción
E1	Si los datos son inválidos.	El sistema indica que la información no es válida y permite nuevamente su captura.
E2	Campo vacío.	El sistema regresa a la pantalla inicial.

Poscondiciones:
El empacador ha ingresado al sistema y se encuentra en la página principal del sistema.

Caso de uso: **2. Salir del sistema**

Actor: Empacador

Descripción: El empacador debe salir del sistema

Precondiciones:

- El empacador debe estar registrado en el sistema.
- El empacador debe haber entrado al sistema.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accesar al menú Salir del sistema.	2	El sistema indica que te desconectará del sistema	E1

Excepciones:

Id	**Nombre**	**Acción**
E1	Si no se ha guardado la información	El sistema indica que necesita guardar la información o cancelar la acción antes de cerrar la sesión

Poscondiciones:

El empacador ha salido del sistema.

Caso de uso: **3. Dar de alta producto**

Actor: Almacenista

Descripción: El almacenista debe dar de alta un producto.

Precondiciones:

- El almacenista debe haber entrado al sistema.
- El almacenista debe tener los permisos necesarios para acceder al menú indicado.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Acceder al menú de captura de producto.	2	Aparece la pantalla de captura de producto.	
3	Captura los datos solicitados en el formato y los guarda.	4	El sistema indica que el producto ha sido guardado.	E1, E2

Excepciones:

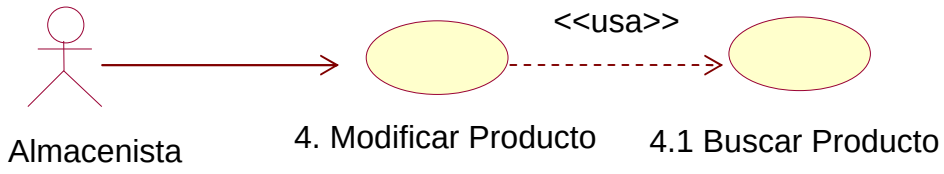
Id	Nombre	Acción
E1	Dato no válido	El sistema indica que el dato no es válido y permite nuevamente la captura.
E2	Campo vacío.	El sistema indica que se deben llenar todos los campos y permite nuevamente la captura.

Poscondiciones:

El almacenista ha dado de alta un producto en el sistema.

Caso de uso: **4. Modificar Producto**

Actor: Almacenista



Descripción: El almacenista realizará una modificación de un producto.

Precondiciones:

- El almacenista debe estar dentro del sistema.
- El almacenista debe tener los permisos necesarios para modificar productos
- El almacenista debe tener el número de producto que necesita modificar.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El almacenista elige la opción para modificar producto.	2	Despliega la pantalla correspondiente.	
3	El almacenista captura el número de producto a modificar.	4	El sistema muestra el producto a modificar.	E1, E2
5	El almacenista modifica los datos del producto y guarda la información.	6	El sistema guarda la información.	

Excepciones:

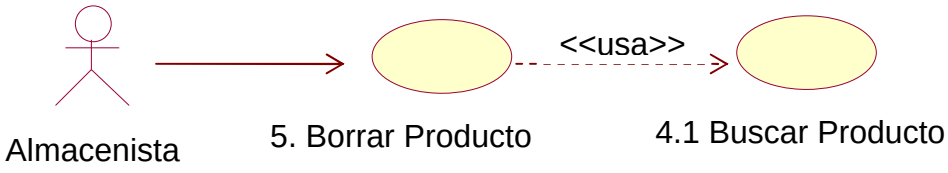
Id	Nombre	Acción
E1	Si el producto no existe	El sistema le informa al almacenista que el producto no existe.
E2	Dato no válido	El sistema indica que no es válido el dato.

Poscondiciones:

- El sistema permite la modificación del producto.

Caso de uso: **5. Borrar Producto**

Actor: Almacenista



Descripción: El almacenista debe borrar.

Precondiciones:

- El almacenista debe estar dentro del sistema.
- El almacenista debe tener los permisos necesarios para dar de baja material.
- El almacenista debe tener el ID del producto que va a borrar

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige del menú la opción para dar de baja el producto.	2	Despliega la pantalla correspondiente.	
3	El almacenista teclea el número de producto que va a dar de baja	4	El sistema indica el producto correspondiente al número introducido por el almacenista.	E1, E2

Excepciones:

Id	Nombre	Acción
E1	Si el número de producto no existe	El sistema informa al almacenista que el producto no existe.

Poscondiciones:

El sistema borra el producto.

Caso de uso: **6. Dar de alta material**

Actor: Almacenista

Descripción: El almacenista debe dar de alta un material.

Precondiciones:

- El almacenista debe haber entrado al sistema.
- El almacenista debe tener los permisos necesarios para acceder al menú indicado.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Acceder al menú de captura de material.	2	Aparece la pantalla de captura de material.	
3	Captura los datos solicitados en el formato y los guarda.	4	El sistema indica que la información ha sido guardada.	E1, E2

Excepciones:

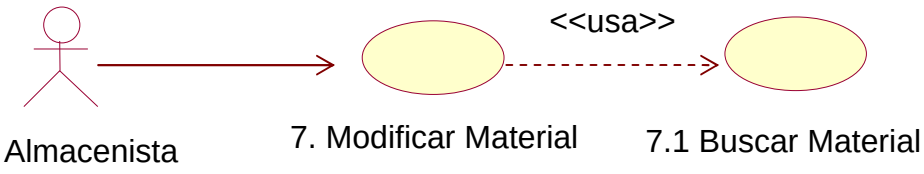
Id	Nombre	Acción
E1	Campo vacío.	El sistema indica que se deben llenar todos los campos y permite nuevamente la captura.

Poscondiciones:

El almacenista ha dado de alta un nuevo material en el sistema.

Caso de uso: **7. Modificar Material**

Actor: Almacenista



Descripción: El almacenista realizará una modificación de un material.

Precondiciones:

- El almacenista debe estar dentro del sistema.
- El almacenista debe tener los permisos necesarios para modificar materiales.
- El almacenista debe tener el número de material que necesita modificar.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El almacenista elige la opción para modificar material.	2	Despliega la pantalla correspondiente.	
3	El almacenista captura el número de material a modificar.	4	El sistema muestra el material a modificar.	E1, E2
5	El almacenista modifica los datos del material y guarda la información.	6	El sistema guarda la información.	

Excepciones:

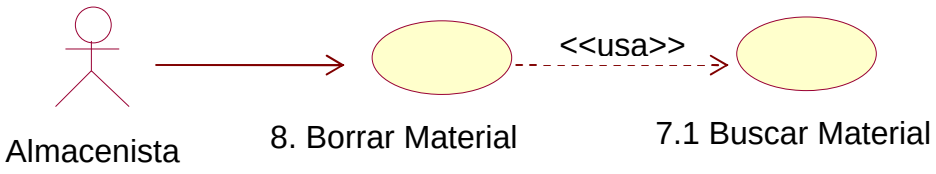
Id	Nombre	Acción
E1	Si el material no existe	El sistema le informa al almacenista que el material no existe.
E2	Dato no válido	El sistema indica que no es válido el dato.

Poscondiciones:

- El sistema permite la modificación del material.

Caso de uso: **8. Borrar Material.**

Actor: Almacenista



Descripción: El almacenista debe borrar un material.

Precondiciones:

- El almacenista debe estar dentro del sistema.
- El almacenista debe tener los permisos necesarios para dar de baja material.
- El almacenista debe tener el ID del material que va a borrar

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige del menú la opción para dar de baja el material.	2	Despliega la pantalla correspondiente.	
3	El almacenista teclea el número de material que va a dar de baja	4	El sistema indica el material correspondiente al número introducido por el almacenista.	E1, E2

Excepciones:

Id	Nombre	Acción
E1	Si el número de material no existe	El sistema informa al almacenista que el material no existe.

Poscondiciones:

El sistema borra el material.

Caso de uso: **9. Dar de alta orden.**

Actor: Planeador

Descripción: El planeador debe dar de alta una orden de empaque.

Precondiciones:

- El planeador debe haber entrado al sistema.
- El planeador debe tener los permisos necesarios para acceder al menú indicado.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Acceder al menú de captura de órdenes.	2	Aparece la pantalla de captura de órdenes.	
3	Captura los datos solicitados en el formato y los guarda.	4	El sistema indica que la orden ha sido guardada.	E1, E2

Excepciones:

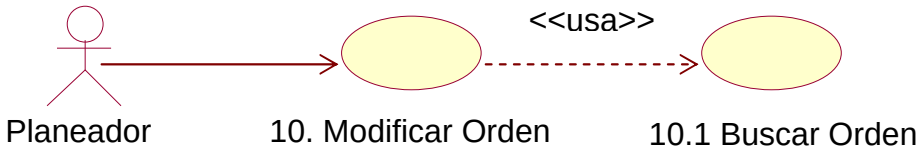
Id	Nombre	Acción
E1	Dato no válido	El sistema indica que el dato no es válido y permite nuevamente la captura.
E2	Campo vacío.	El sistema indica que se deben llenar todos los campos y permite nuevamente la captura.

Poscondiciones:

El planeador ha dado de alta una nueva orden en el sistema.

Caso de uso: **10 Modificar orden.**

Actor: Planeador



Descripción: El planeador realizará una modificación de una orden de empaque.

Precondiciones:

- El planeador debe estar dentro del sistema.
- El planeador debe tener los permisos necesarios para modificar órdenes.
- El planeador debe tener el número de la orden que va a modificar.

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El planeador elige la opción para modificar órdenes.	2	Despliega la pantalla correspondiente.	
3	El planeador captura el número de orden a modificar.	4	El sistema muestra la orden a modificar.	E1, E2
5	El planeador modifica los datos de la orden y guarda la información.	6	El sistema guarda la información.	E3

Excepciones:

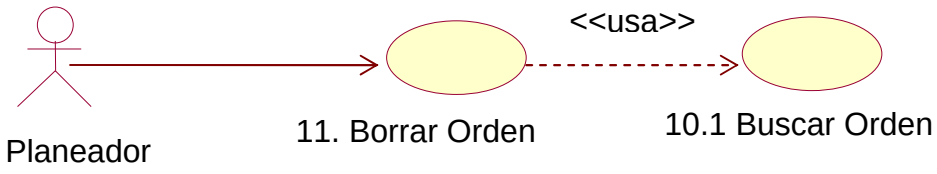
Id	**Nombre**	**Acción**
E1	Si la orden no existe	El sistema le informa al planeador que la orden no existe.
E2	Dato no válido	El sistema indica que no es válido el dato.
E3	Campo vacío	El sistema indica que no se pueden dejar campos vacíos e indica que no guarda los cambios.

Poscondiciones:

El sistema permite la modificación de la orden de empaque.

Caso de uso: **11. Borrar orden**

Actor: Planeador



Descripción: El planeador debe borrar una orden del sistema.

Precondiciones:

- El planeador debe estar dentro del sistema.
- El planeador debe tener los permisos necesarios para dar de baja órdenes.
- El usuario debe tener el número de la orden que va a dar de baja.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige del menú la opción para dar de baja órdenes.	2	Despliega la pantalla correspondiente.	
3	El planeador teclea el número de orden que desea consultar.	4	El sistema indica la orden correspondiente al número introducido por el planeador.	E1

Excepciones:

Id	Nombre	Acción
E1	Si el número de la orden no existe	El sistema informa al planeador que la orden no existe.

Poscondiciones:

El sistema borra la orden.

Caso de uso: **12. Dar de alta usuario**

Actor: Administrador

Descripción: El administrador debe dar de alta un usuario.

Precondiciones:

- El administrador debe haber entrado al sistema.
- El administrador debe tener los permisos necesarios para acceder al menú indicado.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Acceder al menú de captura de usuario.	2	Aparece la pantalla de captura de usuario.	
3	Captura los datos solicitados en el formato y los guarda.	4	El sistema indica que el usuario ha sido guardado.	E1, E2

Excepciones:

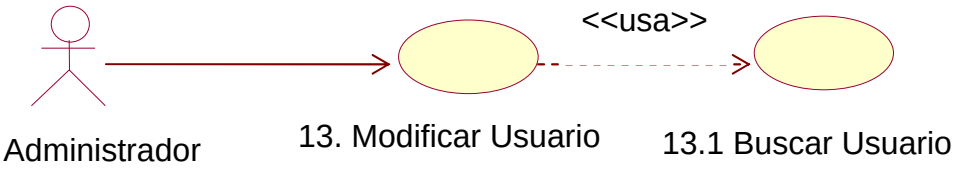
Id	Nombre	Acción
E1	Dato no válido	El sistema indica que el dato no es válido y permite nuevamente la captura.
E2	Campo vacío.	El sistema indica que se deben llenar todos los campos y permite nuevamente la captura.

Poscondiciones:

El administrador ha dado de alta un usuario en el sistema.

Caso de uso: **13. Modificar Usuario.**

Actor: Administrador



Descripción: El administrador realizará una modificación de un usuario

Precondiciones:

- El administrador debe estar dentro del sistema.
- El administrador debe tener los permisos necesarios para modificar productos
- El administrador debe tener el número de usuario que necesita modificar.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El almacenista elige la opción para modificar usuario.	2	Despliega la pantalla correspondiente.	
3	El almacenista captura el número de usuario a modificar.	4	El sistema muestra el usuario a modificar.	E1, E2
5	El almacenista modifica los datos del usuario y guarda la información.	6	El sistema guarda la información.	

Excepciones:

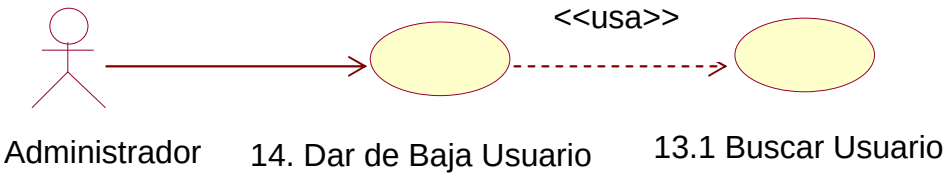
Id	Nombre	Acción
E1	Si el usuario no existe	El sistema le informa al administrador que el usuario no existe.
E2	Dato no válido	El sistema indica que no es válido el dato.

Poscondiciones:

- El sistema permite la modificación del usuario.

Caso de uso: **14. Borrar usuario**

Actor: Administrador



Descripción: El administrador debe dar de baja un usuario del sistema.

Precondiciones:

- El administrador debe estar dentro del sistema.
- El administrador debe tener los permisos necesarios para dar de baja órdenes.
- El administrador debe tener el número de usuario que va a dar de baja.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige del menú la opción para dar de baja usuario.	2	Despliega la pantalla correspondiente.	
3	El planeador teclea el número de usuario que desea consultar.	4	El sistema indica el usuario correspondiente al número introducido por el administrador.	E1

Excepciones:

Id	Nombre	Acción
E1	Si el número de usuario no existe	El sistema informa al administrador que el usuario no existe.

Poscondiciones:

El sistema borra al usuario.

4.7 Diagramas de Análisis.

A continuación se muestran los diagramas de análisis del sistema junto con una breve explicación de cada uno de ellos.

4.7.1 Ordenes

En la figura 4.2 se muestra el diagrama de análisis de órdenes, en este diagrama se representa el módulo de “ordenes”, en el cual, como podemos ver con los estereotipos boundary, que dicho módulo consiste en dar de alta, buscar, modificar y borrar una orden, después de elegir cualquiera de estas opciones, el sistema actúa mediante un estereotipo de control para acceder a la base de datos donde almacena la nueva información que se guarda.

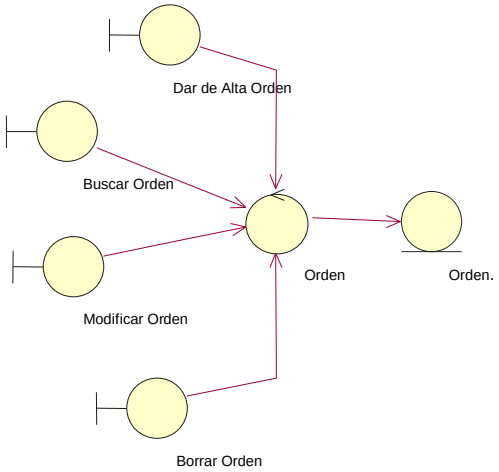


Figura 4.2 Diagrama de análisis órdenes

4.7.2 Productos

En el diagrama de la figura 4.3 se explica el módulo de productos, donde se representan con estereotipos boundary las acciones que se pueden realizar, que son dar de alta producto, buscar producto, modificar producto y dar de baja producto, los cuales dependen de una clase de control para poder realizar determinada actividad y así acceder a la base de datos donde realiza los cambios indicados; busca, da de alta o borra los productos indicados.

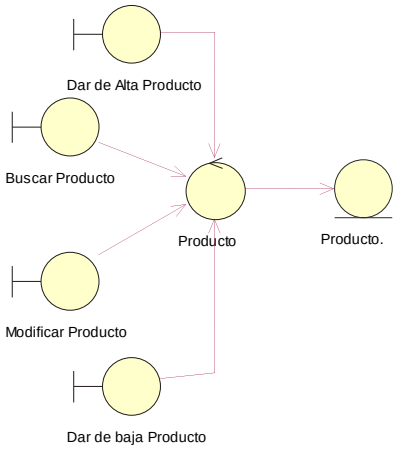


Figura 4.3 Diagrama de análisis Productos

4.7.3 Materiales

El funcionamiento del módulo de materiales es muy similar al de productos y aparece en la figura 4.4, ya que también consta de dar de alta, buscar, modificar y borrar, la diferencia es que en este módulo mediante interfaces de control, se accede a la base de datos en la tabla de materiales y ahí es donde trabaja y hace los cambios ordenados por el usuario.

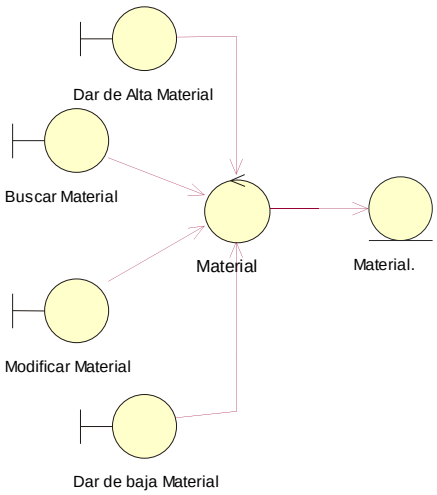


Figura 4.4 Diagrama de análisis Materiales

4.7.4 Usuario

En este módulo, como vemos en la figura 4.5, al igual que los anteriormente nombrados, consta de cuatro diferentes secciones: dar de alta usuario, buscar usuario, modificar usuario y borrar usuario, cuando se requiere hacer cualquiera de estas actividades, por medio de la interfaz de control se puede acceder a la tabla de usuarios que se encuentra dentro de la base de datos del sistema, para realizar las tareas encomendadas por el usuario.

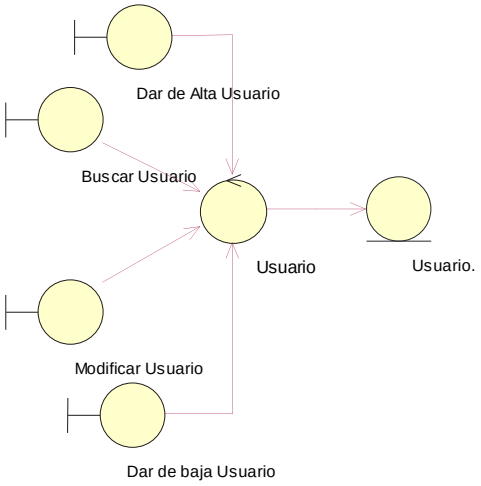


Figura 4.5 Diagrama de análisis Usuario

4.7.5 Entrada al sistema.

En la figura 4.6, tenemos que para entrar al sistema primero mediante una interfaz de control se accede a la tabla de usuarios para poder comprobar que el usuario esté registrado en el sistema y así poder conceder el acceso al mismo.

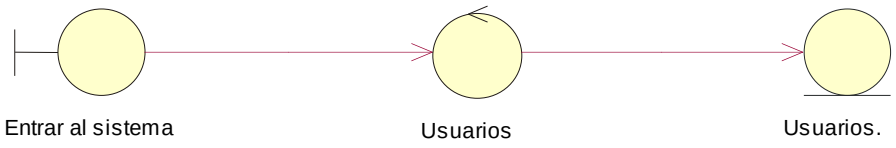


Figura 4.6 Diagrama de análisis Entrada al sistema

5. Diseño del Sistema

En este capítulo incluyo el diseño del sistema en el cual se encuentran los diagramas de clases a bajo nivel, diagrama de clases a alto nivel y diagrama de secuencias.

5.1 Diagrama de clases a bajo nivel.

A continuación se muestra en la figura 5.1 el diagrama de clases a bajo nivel en el cual se puede ver la relación que existe entre las diferentes clases, las cuales tienen relación con la clase orden, ya que esta depende de la información de la Clase Usuario, de la Clase Producto y de la Clase Material, pero la estos tipos de clase no tienen relación entre sí, la clases producto tiene relación con la clase “presentacion” y la clase material tiene relación con la clase “ubicacion”.

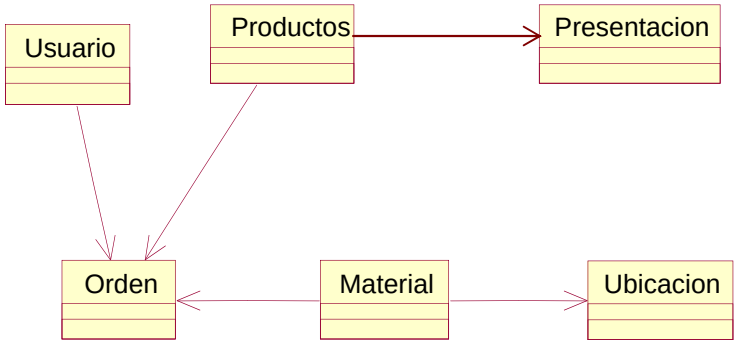


Figura 5.1 Diagrama de clases a bajo nivel

5.2 Diagrama de Clases a Alto Nivel

En el diagrama de Clases a Alto Nivel que se muestra en la figura 5.2, se ve claramente la relación que existe entre cada Clase, incluyendo ya los métodos y atributos que tiene cada una de ellas, se puede ver la relación que tienen la Clase Usuario, la Clase Producto y la Clase material con la Clase Orden que son los métodos idUsuario, idproducto y inmaterial, además la clase presentación tiene relación con la clase productos y la clase ubicación tiene relación con la clase material.

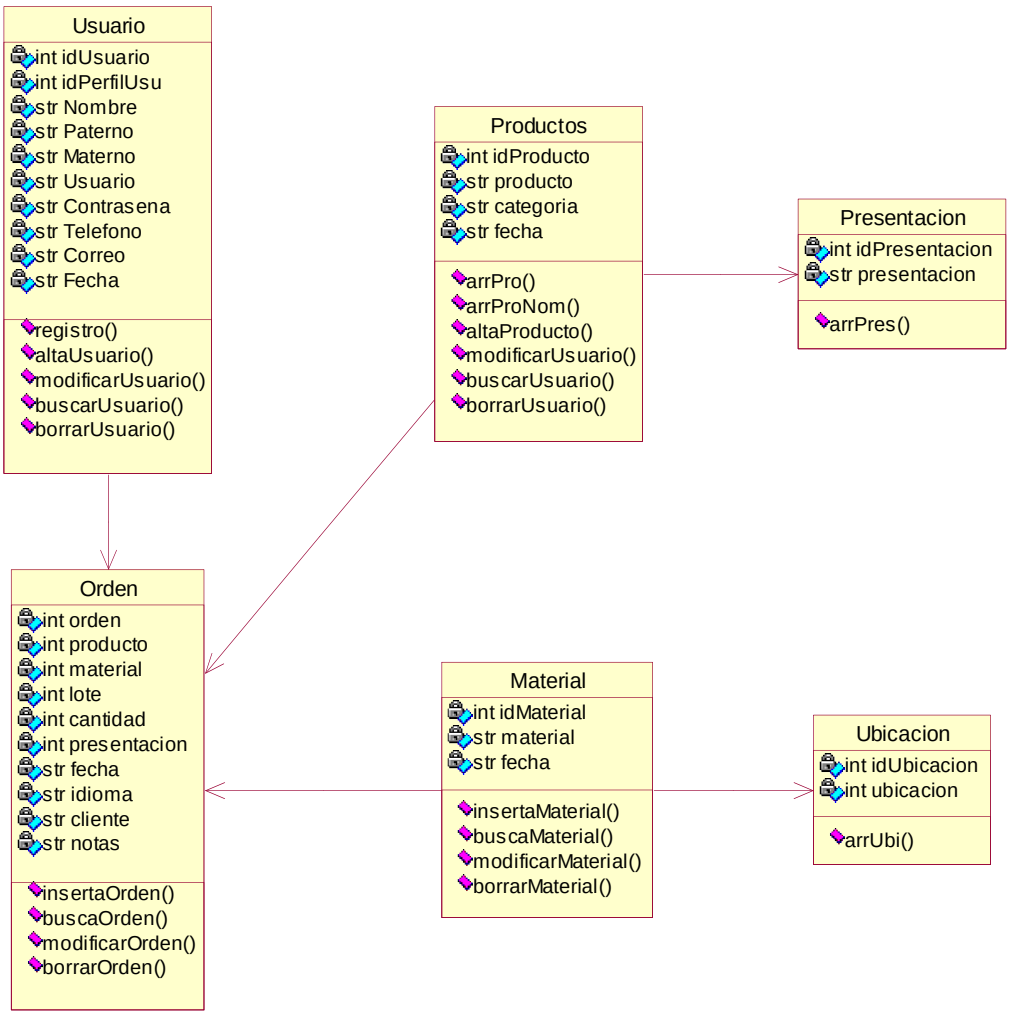


Figura 5.2 Diagrama de clases a Alto nivel

5.3 Diagramas de Secuencia.

Como parte de la descripción del comportamiento de este sistema, he realizado los diagramas de secuencia pertinentes en los que se muestra la interacción de los actores con el sistema, en la que dichos actores generan eventos solicitando al sistema operaciones.

Los diagramas están acomodados en el orden de los casos de uso del diagrama general que se encuentra en el capítulo cuatro.

Dar de Alta Producto.

Para dar de alta un producto el almacenista selecciona dar de alta Producto y se abre la Pantalla de Producto, después captura la información requerida y valida los datos, la clase producto guarda la información dentro de la tabla Producto. Ver figura 5.3.

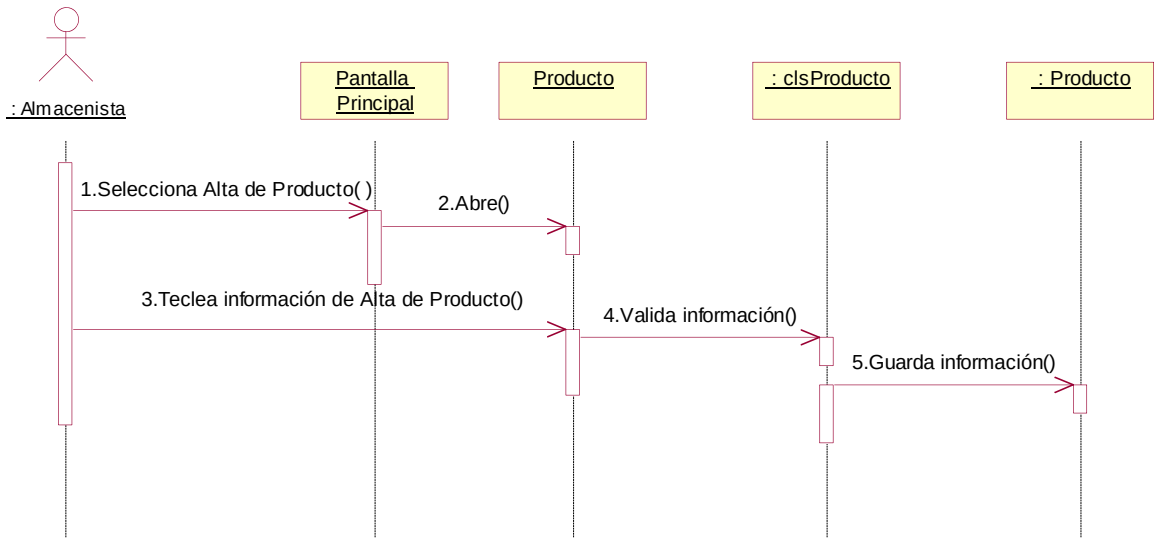


Figura 5.3 Diagrama de Secuencia Dar de alta producto

Modificar Producto.

En la figura 5.4 se muestra el diagrama de secuencia Modificar Producto. Cuando el almacenista desea modificar un producto selecciona desde la pantalla principal Buscar Producto, una vez que se abre la pantalla de Búsqueda, el almacenista captura la información necesaria para buscar el producto a modificar y la valida, la clase producto busca la información en la tabla Producto, regresa y muestra las coincidencias, el almacenista selecciona el producto a modificar y abre la pantalla, después modifica los campos deseados y valida la información; por último la clase Producto guarda los cambios dentro de la tabla Producto.

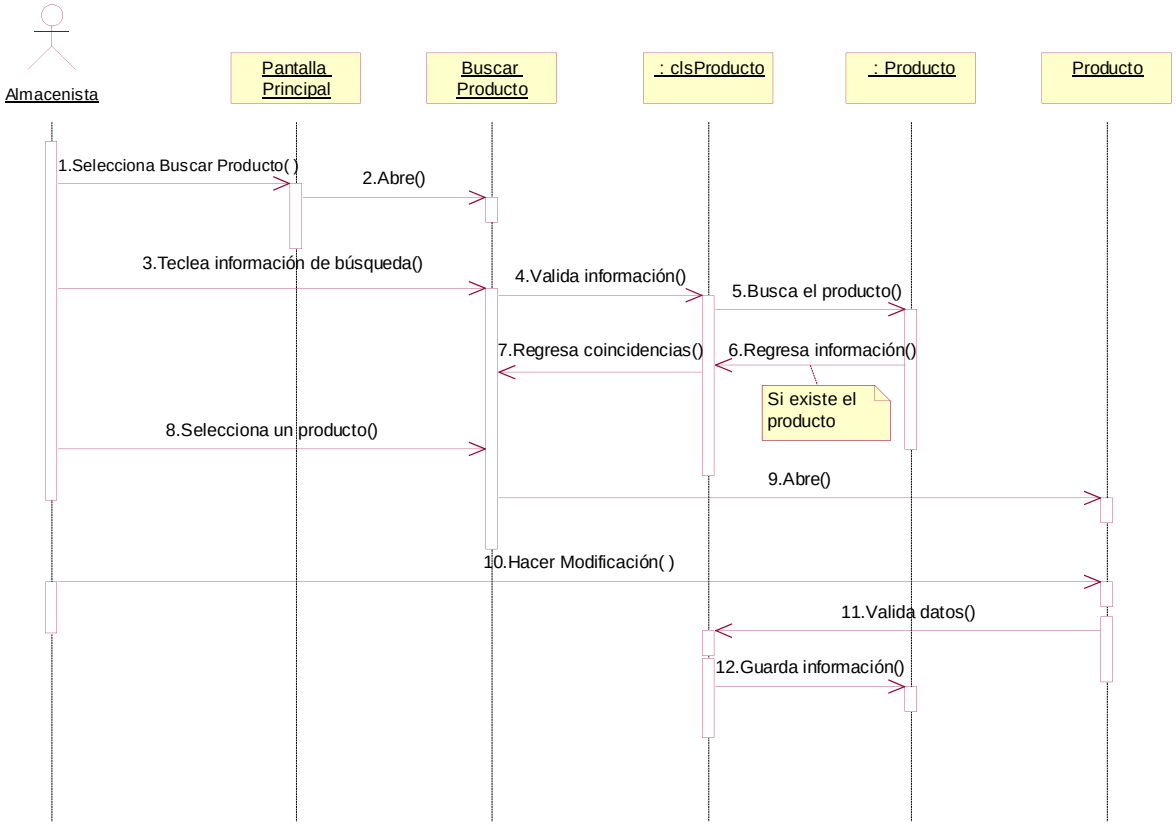


Figura 5.4 Diagrama de secuencia Modificar Producto

Borrar Producto.

El almacenista selecciona Buscar producto y se abre la pantalla de búsqueda de Producto, después teclea la información necesaria y la valida, la clase Producto busca la información en la tabla Producto y regresa las coincidencias, el almacenista selecciona el producto deseado y lo da de baja; finalmente la clase Producto elimina el producto, esto se puede ver en la figura 5.5.

Figura 5.5 Diagrama de secuencia Borrar Producto

Dar de Alta Material.

En el diagrama de la figura 5.6, el almacenista selecciona Dar de alta Material, se abre la pantalla de Material y teclea la información de Alta, después valida los datos y la clase Material guarda la información dentro de la tabla Material.

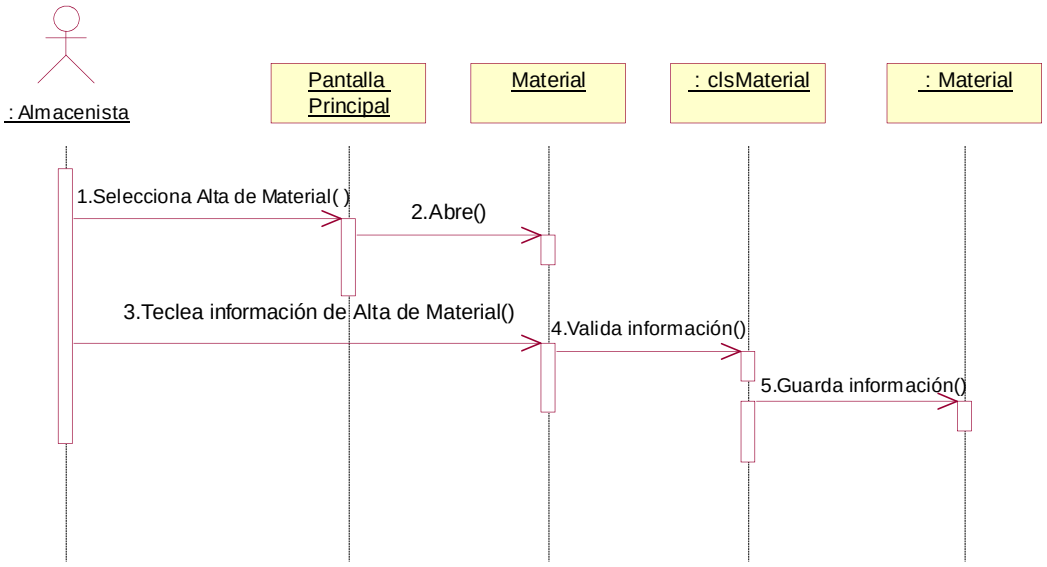


Figura 5.6 Diagrama de secuencia Dar de alta Material

Modificar Material.

Para modificar un material, en la figura 5.7 se muestra que el almacenista selecciona Buscar Material, una vez abierta dicha pantalla, teclea los parámetros de búsqueda y valida la información, la clase Material busca el material dentro de la tabla Material, regresa y muestra las coincidencias si es que existen, el almacenista selecciona el material que desea modificar, realiza los cambios deseados y valida los datos; después la clase Material guarda los cambios dentro de la tabla Material.

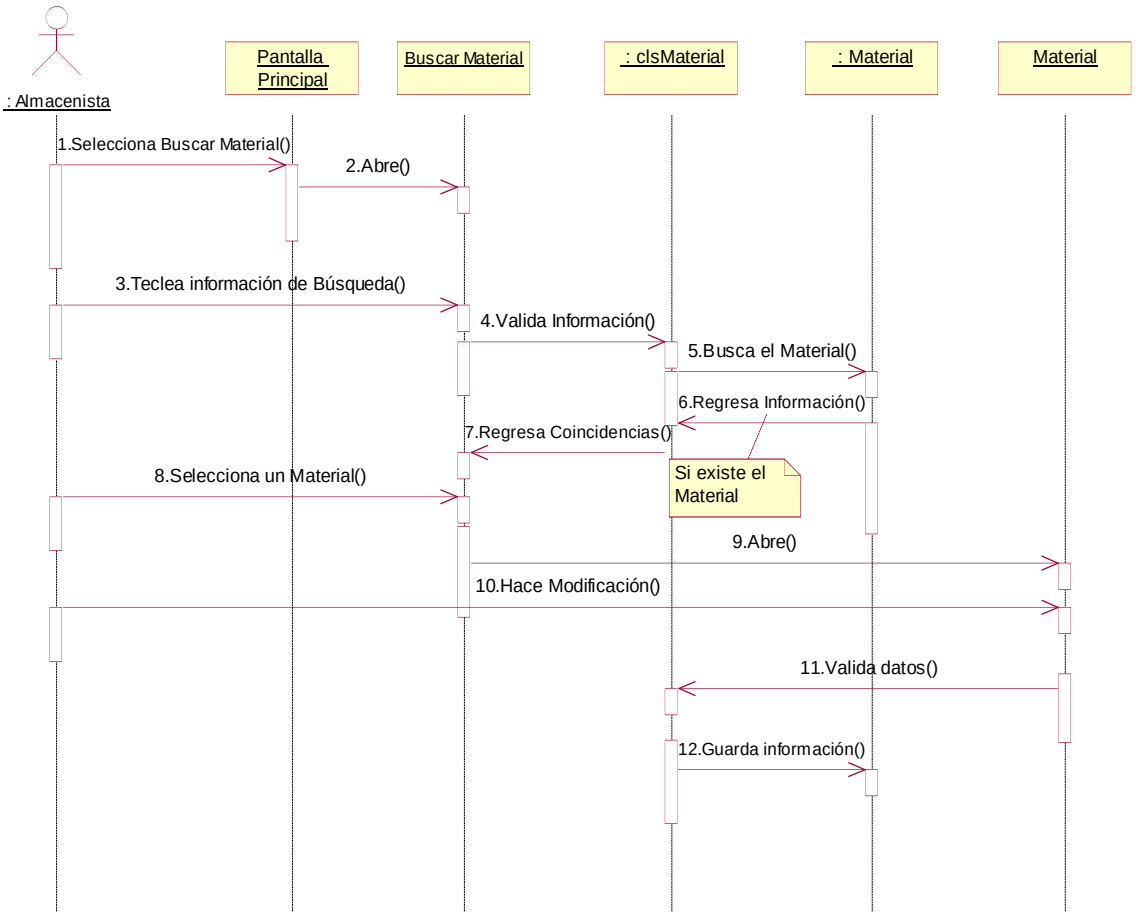


Figura 5.7 Diagrama de secuencia Modificar Material

Borrar Material.

Cuando el almacenista desea dar de baja un Material selecciona Buscar material, se abre la pantalla indicada y teclea los datos necesarios para poder realizar la búsqueda y valida la información, la clase Material busca el material coincidente dentro de la tabla Material, regresa la información encontrada y la muestra en pantalla, el almacenista selecciona el material deseado y lo da de baja; la clase Material elimina el material de la tabla. Figura 5.8.

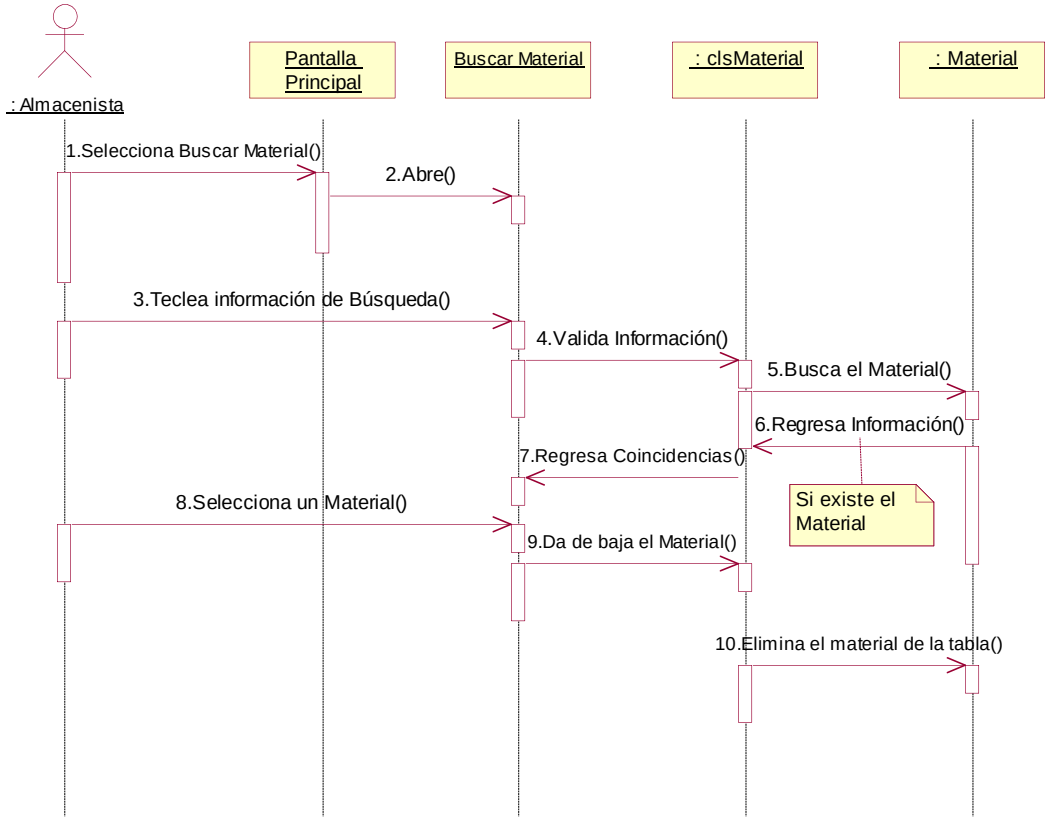


Figura 5.8 Diagrama de secuencia Borrar Material

Dar de Alta Orden.

El planeador selecciona Dar de Alta Orden, se abre la pantalla Orden, teclea la información requerida para dar de alta la orden y valida la información; finalmente la Clase Orden guarda la información dentro de la tabla Orden. Ver figura 5.9.

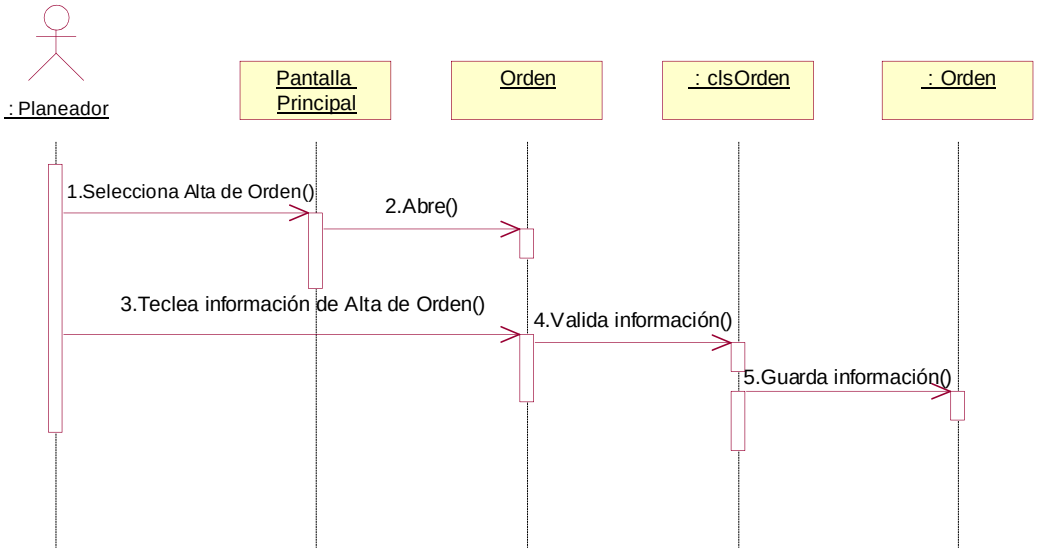


Figura 5.9 Diagrama de secuencia Dar de alta orden

Modificar Orden.

En caso de que el planeador desee modificar una orden selecciona desde la pantalla principal Buscar Orden, se abre la pantalla de búsqueda, teclea la información requerida y valida los datos, la clase orden busca los datos dentro de la tabla Orden, regresa y muestra en pantalla las coincidencias encontradas, el planeador selecciona la orden a modificar, realiza los cambios deseados y valida la información; la Clase Orden guarda los datos dentro de la tabla Orden. La Figura 5.10 muestra el diagrama de secuencia.

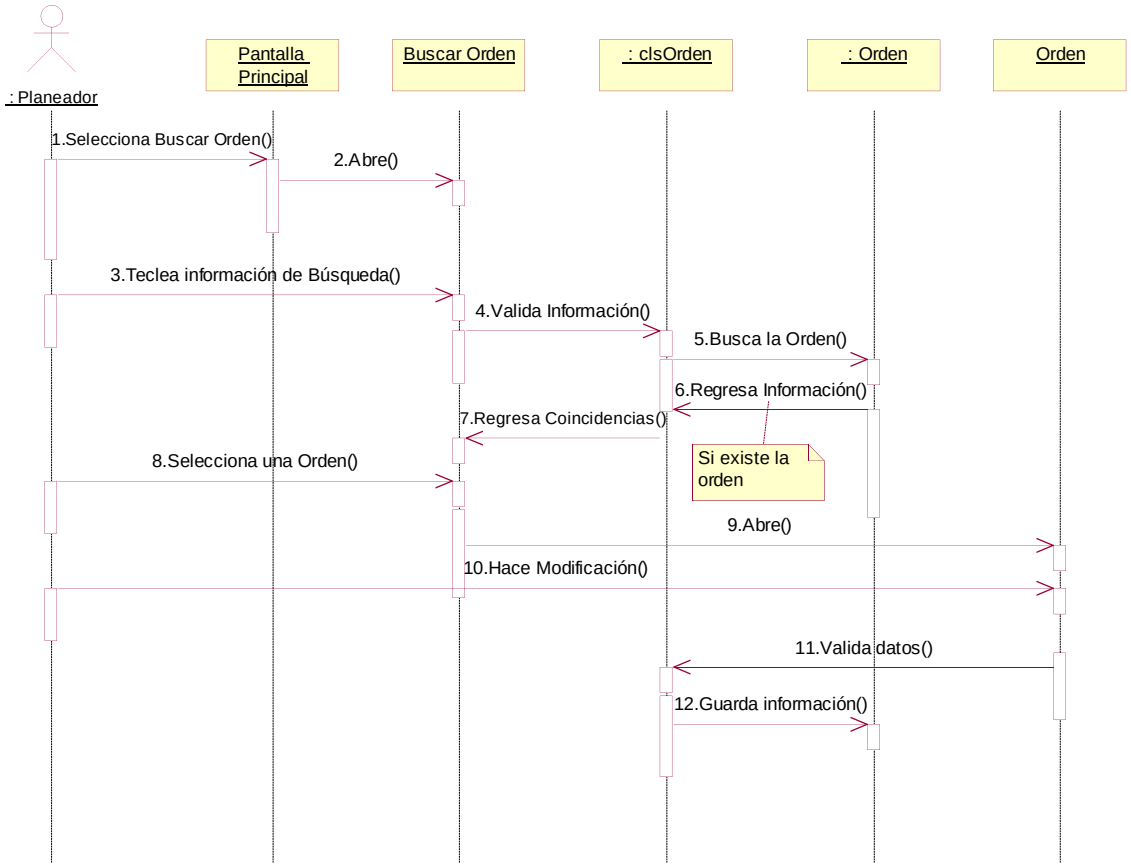


Figura 5.10 Diagrama de secuencia Modificar Orden

Borrar Orden.

Para dar de baja la orden el planeador selecciona Buscar Orden y cuando se abre dicha pantalla captura los datos necesarios, valida la información y la clase Orden busca los datos dentro de la tabla Orden, después regresa las coincidencias y las muestra en pantalla, el planeador selecciona la orden que desea dar de baja y la elimina; la Clase Orden realiza la eliminación en la tabla Orden. En la figura 5.11 se ve con mayor claridad.

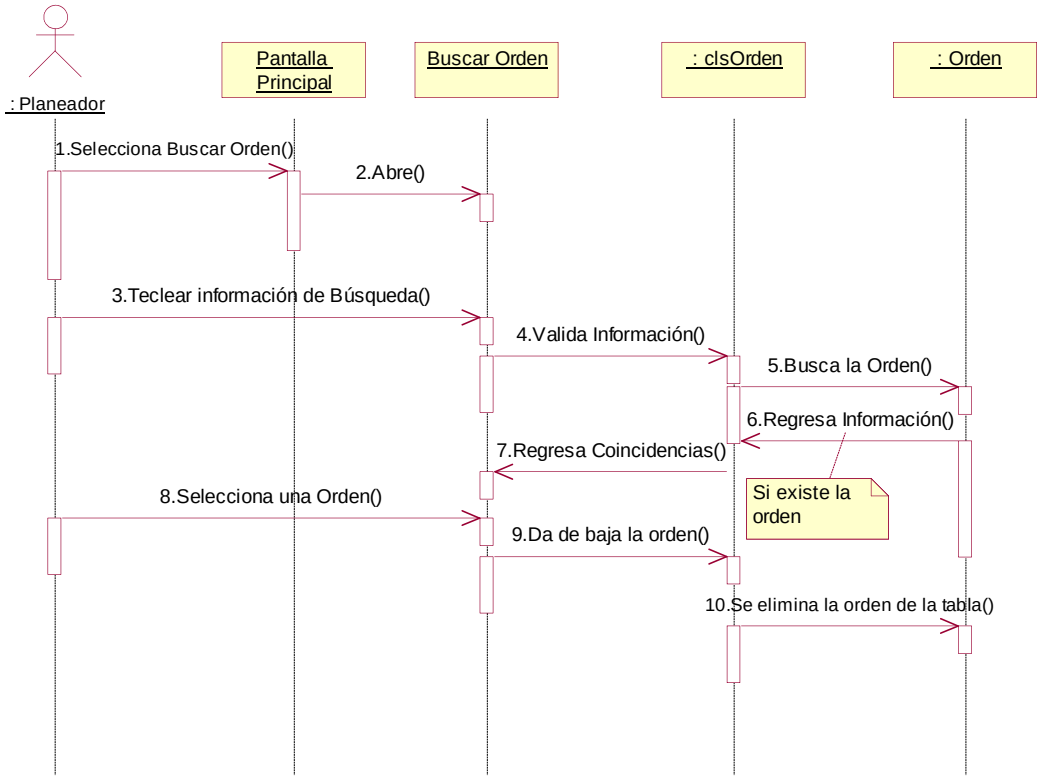


Figura 5.11 Diagrama de secuencia Borrar Orden

Dar de Alta Usuario.

En la figura 5.12 tenemos en diagrama de secuencia Dar de Ata Usuario, en el cual el administrador selecciona Alta de usuario y una vez abierta la pantalla requerida teclea la información de Alta y la valida; la Clase Usuario guarda la información dentro de la tabla Usuario.

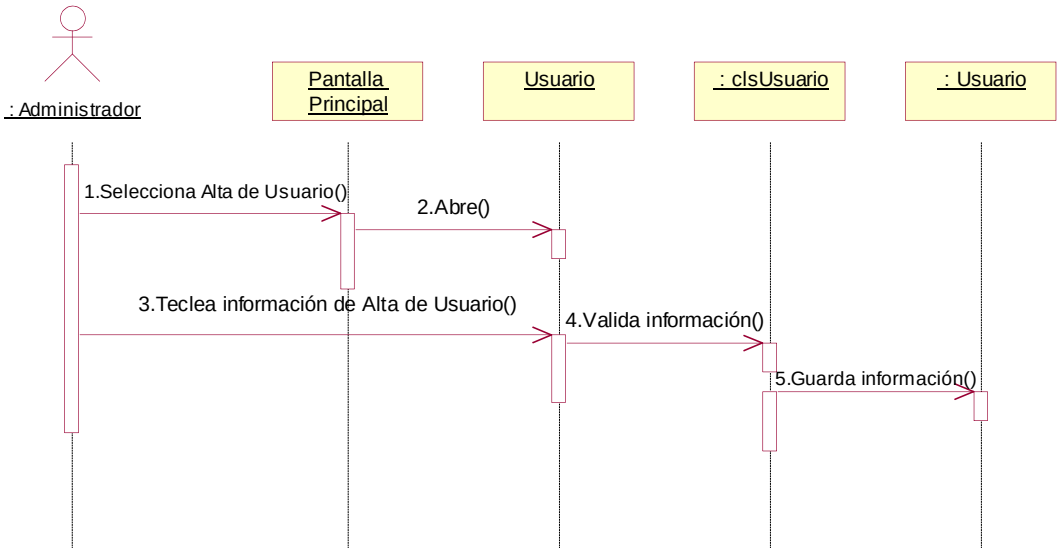


Figura 5.12 Diagrama de secuencia Dar de alta Usuario

Modificar Usuario.

En el diagrama de secuencia de la figura 5.13, el administrador selecciona Buscar Usuario, se abre la pantalla de búsqueda y teclea los parámetros requeridos para realizar la búsqueda, después valida los datos y la clase Usuario los busca dentro de la tabla Usuario y si encuentra coincidencias regresa la información y muestra una lista en pantalla, el administrador selecciona el usuario a modificar y realiza los cambios pertinentes, valida la información y la Clase Usuario guarda los cambios en la tabla Usuario.

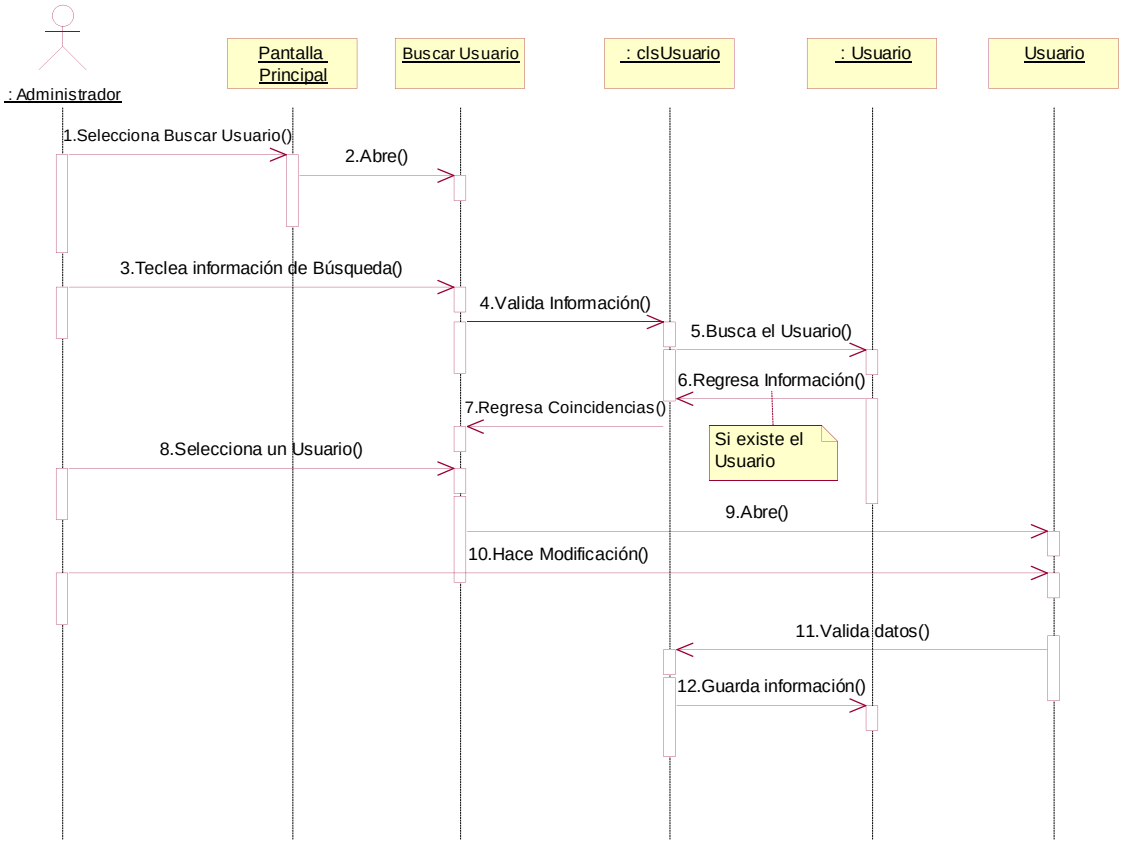


Figura 5.13 Diagrama de secuencia Modificar Usuario

Borrar Usuario.

En este caso de uso el administrador selecciona Buscar Usuario, se abre la pantalla de búsqueda y teclea los parámetros de búsqueda de usuario y la valida, la clase Usuario busca los datos coincidentes dentro de la tabla Usuario, si existen resultados los regresa y los muestra en pantalla, el administrador selecciona el usuario a eliminar y lo da de baja; la Clase Usuario hace la eliminación de la tabla Usuario. Ver figura 5.14.

Figura 5.14 Diagrama de secuencia Borrar Usuario

6. Implementación y Pruebas.

En este capítulo se muestran las pantallas del sistema con una breve explicación de cada una de ellas, así como también el código de uno de los módulos de dicho sistema. he elegido explicar el módulo principal que es el de órdenes, ya que los demás son similares.

6.1 Pantallas del Sistema.

Comenzaremos por mostrar las pantallas y la explicación de las mismas para un mejor entendimiento.

6.1.1 Ordenes.

Dar de Alta orden.

La función de esta pantalla será dar de alta una orden de empaque, en donde el número de orden es auto numérico, se capturará la fecha de creación de la orden, el producto que se va a empacar se elegirá de la lista, así como el material que se utilizará para empacar el producto, elegir el idioma en que se harán las etiquetas y la unidad de medida, y se capturará la cantidad a empacar, el número de lote y algún comentario u observación que tenga que hacerse a los que participen en el proceso de empaque. Finalmente, se tiene acceso a buscar una orden y a regresar a principal con sólo ir a la parte superior, y en la parte inferior se tendrá acceso a los demás módulos que existen, como son productos, materiales, órdenes y usuarios. La pantalla se muestra en la figura número 6.1.1

Figura 6.1.1 Pantalla Dar de alta orden

Buscar Orden.

En esta pantalla (figura 6.1.2) se tiene la opción para elegir la forma de búsqueda, si se desea buscar por número de orden, por fecha, lote o por producto; de esta manera se puede encontrar más fácilmente la orden requerida. También se tiene la opción de elegir lo que se desea hacer con las órdenes encontradas con los datos que se han introducido, si se quiere modificar o eliminar, en caso de que se elija Modificar nos da acceso a la siguiente pantalla que es Modificar Orden. En caso de que el usuario desee realizar otra operación, en la parte inferior de la pantalla se tiene la opción de ingresar a dar de alta orden y pantalla principal, y en la parte inferior se tiene la opción de ir a orden, producto, material y usuario.

Figura 6.1.2 Pantalla Buscar orden

Modificar Orden.

Una vez que se ha seleccionado en la pantalla de búsqueda la orden que se desea modificar, entonces nos aparece la pantalla de modificar orden, en donde se da acceso a modificar los datos necesarios y una vez hecho esto, se tiene la opción de guardar o cancelar la modificación. En la parte inferior se puede ingresar a las pantallas de materiales, usuarios, órdenes y productos. En la parte superior se tiene acceso al menú de buscar orden, alta de orden y pantalla principal. (Figura 6.1.3).

Figura 6.1.3 Pantalla Modificar orden

6.1.2 Productos.

Dar de Alta Producto.

En la figura 6.1.4 se muestra la pantalla para dar de alta un producto, primero hay que escribir el id del producto, después escribir el nombre del producto, la fecha en que se está realizando el alta, especificar de qué categoría es (sal, ácido y solvente), seleccionar las medidas en que se maneja dicho producto. También se tiene la opción de ir a buscar un producto o regresar a la pantalla principal en la parte superior, y en la parte inferior se tiene acceso a ir a las pantallas de órdenes, productos, materiales y usuarios.

Figura 6.1.4 Pantalla Dar de alta producto

Buscar Producto.

Cuando el usuario desee buscar un producto (figura 6.1.5), se tienen dos opciones, se puede buscar por número de producto (que es único) o por nombre del producto, inmediatamente después de oprimir el botón de buscar, aparecerá en la parte de abajo los resultados de la búsqueda, donde se tendrá la opción de ingresar al producto encontrado para modificarlo o simplemente eliminarlo. En caso de que el usuario necesite realizar otra operación dentro del módulo de producto, tendrá acceso a alta de producto y a pantalla principal desde la parte superior de la pantalla, en caso de que desee utilizar otro módulo del sistema, podrá hacerlo desde la parte inferior de la pantalla.

Figura 6.1.5 Pantalla Buscar Producto

Modificar Producto.

En caso de haber seleccionado en los resultados de la búsqueda la opción de modificar, se tendrá acceso a la pantalla de Modificar Producto como se muestra en la figura 6.1.6, en la cual se tendrá acceso a realizar las modificaciones necesarias al producto seleccionado. Al igual que las otras pantallas, se tendrá la opción de acceder desde esta pantalla a dar de alta o buscar producto, a la pantalla principal, y a los módulos de orden, material, producto y usuario.

Figura 6.1.6 Pantalla Modificar Producto

6.1.3 Material.

Dar de Alta Material.

En esta pantalla (figura 6.1.7), se captura la fecha en que se está haciendo la captura, el número del material, el nombre del material y la ubicación en que se encuentra almacenado, se tiene la opción de ir a la pantalla de buscar material, de regresar a la pantalla principal o de ir a los módulos de orden, material, producto o usuario.

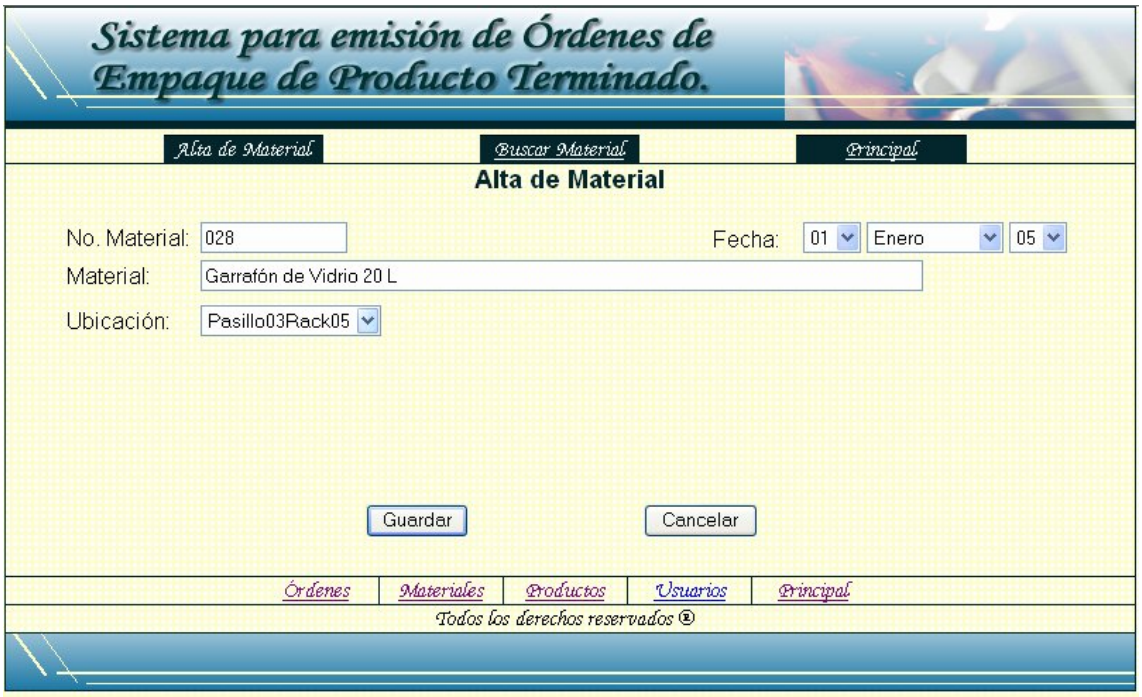


Figura 6.1.7 Pantalla Dar de alta Material

Buscar Material.

Para buscar un material se tienen dos opciones, por número de material y por nombre del material. Después de oprimir el botón de buscar, aparecen en la parte de abajo los resultados de la búsqueda, cada uno con la opción de modificar o dar de baja. Esta pantalla se muestra en la figura 6.1.8.

Figura 6.1.8 Pantalla Buscar Material

Modificar Material.

Para modificar el material, después de haber seleccionado la opción de modificar, aparece una pantalla para realizar todos los cambios que el usuario considere necesarios (figura 6.1.9). Esta pantalla tiene la opción de de ir a dar de alta, buscar o ir a pantalla principal. También se tiene la facilidad de ir a los módulos de orden, usuario, material y producto.

Figura 6.1.9 Pantalla Modificar Material

6.1.4 Usuario.

Dar de Alta Usuario.

Para dar de alta un usuario el id de Usuario será auto numérico, la fecha será en la que se está dando de alta el usuario, se capturará el nombre, apellido paterno, apellido materno, el usuario, la contraseña, la validación de la contraseña, el teléfono del usuario, su correo electrónico y finalmente el perfil de usuario. Se tendrá la facilidad de entrar a la página de buscar usuario o ir a principal y también de ir al módulo de material, orden, usuario y producto. Figura 6.1.10

Figura 6.1.10 Pantalla Dar de alta usuario

Buscar Usuario.

Cuando se desea buscar un usuario (figura 6.1.11), se tienen dos opciones en la pantalla, se puede hacer por id de usuario o por nombre de usuario, después de seleccionar el botón de buscar, aparecerán los resultados de la búsqueda y cada uno de estos tendrá la opción de modificar o eliminar el usuario señalado. Se tiene la facilidad de ir a dar de alta al usuario, a principal y de ir a producto, usuario, material y orden si así se desea.

Figura 6.1.11 Pantalla Buscar Usuario

Modificar Usuario.

Una vez seleccionada la opción de modificar usuario, se tiene acceso a una pantalla donde se tiene la facilidad de realizar todos los cambios necesarios con opción a guardar la información o cancelarla como se muestra en la figura 6.1.12. Al igual que todas las demás pantallas, la opción de cambiar de pantalla a dar de alta un usuario, buscarlo, o ir a principal, así como acceder a los módulos de usuario, material, orden y producto.

Figura 6.1.12 Pantalla Modificar Usuario

6.2 Código

En este capítulo explicaré el código del sistema, sólo explicaré la conexión y el módulo de órdenes, ya que para los demás módulos es muy similar.

Separé los códigos de acuerdo al patrón de diseño utilizado, que como ya lo había mencionado es Modelo Vista Controlador.

6.2.1 Modelo

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones, no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos.

6.2.1.1 Conexión

Para realizar la conexión a la base de datos de Mysql se utilizó el siguiente código, en el cual se está mandando a llamar al localhost, la base de datos de nombre ordenes, el usuario “user” y el password “user”.

```
public class Conexion {

   public Conexion() {
   }

   public static Connection  getConexion(){
   Connection cn = null;
      try{
         Class.forName("com.mysql.jdbc.Driver");
         cn=DriverManager.getConnection("jdbc:mysql://localhost/ordenes","user","user");
      }catch(ClassNotFoundException e ){
         System.out.println("No se encuentra el driver de la Base de Datos");
      }catch(SQLException e){
         System.out.println(e.toString());
         System.out.println("Error en SQL");
      }

   return cn;
   }
```

En esta parte lo que se está haciendo es cerrar la conexión a la base de datos, éste paso es necesario, ya que de otro modo consumiría recursos de memoria y alentaría el sistema.

```
  public static void cerrarConexion(Object obj ){
    try{
          if(obj instanceof Connection){
             ((Connection)obj).close();

          if(obj instanceof Statement){
           ((Statement)obj).close() ;
          }
          }

      }catch(SQLException e){

            System.out.println("Error :" + e.getMessage() );
      }
  }
```

```
    public static void main(String args[]){
        Connection cnn=null;
        try{
            cnn = Conexion.getConexion();
            System.out.println("details"+cnn.getMetaData() );
            System.out.println(""+cnn.getCatalog());
            System.out.println(""+cnn.getMetaData().getDriverVersion());

        }catch(Exception e){
        }
    }
}
```

6.2.1.2 Orden

En la parte del modelo, la clase orden se divide en dos, Orden y DAOOrden, en la primera se realiza el constructor y se declaran los atributos que contendrá la clase, se declaran los get y set de cada uno de los atributos, a continuación se muestran sólo algunos de los atributos con los que cuenta el constructor.

```
package com.diploweb.model;

public class Orden {
    private int orden;
    private int producto;
    private int material;
    private int lote;
    private int cantidad;
    private int presentacion;
    private String fecha;
    private String idioma;
    private String cliente;
    private String notas;

    public Orden() {
    }
    public Orden(int orden, int producto, int material, int lote, int cantidad,
        int presentacion, String fecha, String idioma, String cliente, String notas){
            this.orden = orden;
            this.producto = producto;
            this.material = material;
            this.lote = lote;
            this.cantidad = cantidad;
            this.presentacion = presentacion;
            this.fecha = fecha;
            this.idioma = idioma;
            this.cliente = cliente;
            this.notas = notas;
    }

    public int getOrden(){
        return orden;
    }
    public int getProducto(){
        return producto;
    }
    public int getMaterial(){
        return material;
    }
    public int getLote(){
        return lote;
    }
    public int getCantidad(){
```

```
        return cantidad;
    }
    public int getPresentacion(){
        return presentacion;
    }
    public String getFecha(){
        return fecha;
    }
    public String getIdioma(){
        return idioma;
    }
    public String getCliente(){
        return cliente;
    }
    public String getNotas(){
        return notas;
    }


    public void setOrden(int orden){
        this.orden = orden;
    }
    public void setProducto(int producto){
        this.producto = producto;
    }
    public void setMaterial(int material){
        this.material = material;
    }
    public void setLote(int lote){
        this.lote = lote;
    }
    public void setCantidad(int cantidad){
        this.cantidad = cantidad;
    }
    public void setPresentacion(int presentacion){
        this.presentacion = presentacion;
    }
    public void setFecha(String fecha){
        this.fecha = fecha;
    }
    public void setIdioma(String idioma){
        this.idioma = idioma;
    }
    public void setCliente(String cliente){
        this.cliente = cliente;
    }
    public void setNotas(String notas){
        this.notas = notas;
    }

}
```

6.2.1.3 DAO Orden

En esta clase se manda llamar la conexión a la base de datos, y también se cierra dicha conexión, se declaran los métodos como son insertaOrden, se crea el objeto o de tipo Orden y se asignan los atributos declarados en la clase Orden, a continuación se muestra sólo un poco de código.

```
package com.diploweb.model;
import java.sql.SQLException;
import java.sql.*;
import java.util.*;

public class DAOOrden {

   public DAOOrden() {
   }
   public static Orden insertaOrden(Orden o){
      int iMax = 0;
      String query;
      Connection con=null;
      con=Conexion.getConexion();
      if (con==null){
         return null;
      }
   try{
      Statement stmt = con.createStatement();
      query="INSERT INTO orden(idOrden, idProducto, idMaterial, Lote, Cantidad, idPresentacion, " +
            "FechaCreacion, Idioma, Cliente) values ("+
            o.getOrden() +","+ o.getProducto() +","+ o.getMaterial() +","+ o.getLote() +","+
            o.getCantidad()+","+ o.getPresentacion()+","'"+o.getFecha()+"'","'"+ o.getIdioma() +"'","'"+
            o.getCliente()+"'");" ;
      stmt.executeUpdate(query);

      iMax = DAOOrden.maxOrden();
      o.setOrden(iMax);
      stmt.close();
   }catch(SQLException e){
     System.out.println("Error en insertar:" + e.getMessage() );
     return null;
   }
   Conexion.cerrarConexion(con);
   return o;
   }

   public static int maxOrden(){
      String query;
      Connection con=null;
      con=Conexion.getConexion();
      int iMax = 0;
   try{
      Statement stmt = con.createStatement();
      query="Select max(idOrden) from orden;";
      ResultSet rs = stmt.executeQuery(query);
      while(rs.next()){
         iMax = rs.getInt(1);
      }
      stmt.close();
   }catch(SQLException e){
     System.out.println("Error en insertar:" + e.getMessage() );
     return 0;
   }
   Conexion.cerrarConexion(con);
   return iMax;
   }
    public static ArrayList buscaOrden(int idOrden){
```

```
        String query;
        Connection con=null;
        con=Conexion.getConexion();
        ArrayList arr = new ArrayList();
    try{
        Statement stmt = con.createStatement();
        query = "Select * from orden where idOrden=" + idOrden;
        ResultSet rs = stmt.executeQuery(query);
        while(rs.next()){
            arr.add(new Orden(rs.getInt(1), rs.getInt(2), rs.getInt(3), rs.getInt(4), rs.getInt(5), rs.getInt(6),
rs.getString(7),
                    rs.getString(8), rs.getString(9), rs.getString(10)));
        }
    }catch(SQLException e){
       System.out.println("Error en la búsqueda:" + e.getMessage() );
       return null;
    }
    Conexion.cerrarConexion
           (con);
    return arr;
    }
    public static ArrayList buscaOrden(String fechaInicio, String fechaFin){
        String query;
        Connection con=null;
        con=Conexion.getConexion();
        ArrayList arr = new ArrayList();
    try{
        Statement stmt = con.createStatement();
        query = "Select * from orden where fechaCreacion between '" + fechaInicio + "' and '" + fechaFin + "'";
        ResultSet rs = stmt.executeQuery(query);
        while(rs.next()){
            arr.add(new Orden(rs.getInt(1), rs.getInt(2), rs.getInt(3), rs.getInt(4), rs.getInt(5), rs.getInt(6),
rs.getString(7),
                     rs.getString(8), rs.getString(9), rs.getString(10)));
        }
    }catch(SQLException e){
       System.out.println("Error en la búsqueda:" + e.getMessage() );
       return null;
    }
    Conexion.cerrarConexion(con);
    return arr;
    }
}
```

6.2.2 Vista

Esta parte es la que permite la interacción del usuario con el sistema, consta de las pantallas y todo lo que se encuentra a la vista de usuario.

6.2.2.1 AltaOrden

Primero que nada, en este archivo se importan los paquetes que son necesarios para que la pantalla funcione adecuadamente, también se puede ver todo lo que es el código HTML, se delimitan las tablas, el número de celdas y columnas así como su tamaño, se crean los formularios en los que aparecen las cajas de texto, botones y nombres de los campos, para que el usuario pueda dar de alta una orden con los datos correspondientes.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="com.diploweb.model.Productos" %>
<%@page import="com.diploweb.model.Materiales" %>
<%@page import="com.diploweb.model.Presentacion" %>
<%@page import="com.diploweb.model.DAOProductos" %>
<%@page import="com.diploweb.model.DAOMateriales" %>
<%@page import="com.diploweb.model.DAOOrden" %>
<%@page import="com.diploweb.model.DAOPresentacion" %>
<%@page import="com.diploweb.model.DAOUbicacion" %>
<%@page import="java.util.ArrayList" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <link href="<%=request.getContextPath() %>/vista/Estilos.css" type="text/css" rel="stylesheet" />
       <script src="clsUtilerias.js" type="text/javascript"></script>
      <title></title>
   </head>
   <body background="<%=request.getContextPath() %>/vista/img/FONDO.PNG">
      <div align="center">
         <%@include file="menu.jsp"%>
         <form method="POST" action="<%=request.getContextPath()%>/AddOrden">
         <table width="800px">
            <tr class="Labels">
               <td colspan="6"> </td>
            </tr>
            <tr class="Labels">
               <td></td>
               <td>
               </td>
               <td></td>
               <td>Producto:
                  <select id="cboProducto" name="cboProducto">
                     <%
                       Productos pro = new Productos();
                       ArrayList arrPro=DAOProductos.arrPro();
                       if(arrPro !=null){
                         for(int i=0;i<arrPro.size();i++){
                         pro=(Productos)arrPro.get(i);
                         out.write("<option value="+pro.getidProducto() + ">"+ pro.getproducto() +"</option>)");
                         }
                       }
                     %>
                  </select>
```

```
            </td>
            <td></td>
            <td>Fecha:</td>
            <td></td>
          </tr>
          <tr class="Labels">
            <td></td>
            <td>No.Lote: <input type="text" id="txtLote" name="txtLote">
            </td>
            <td></td>
            <td>Cantidad: <input type="text" id="txtCantidad" name="txtCantidad">
            </td>
            <td></td>
            <td>Unidad de Medida:
              <select id="cboUnidad" name="cboUnidad">
                <%
                  Presentacion pres = new Presentacion();
                  ArrayList arrPres=DAOPresentacion.arrPres();


                  if(arrPres !=null){
                    for(int i=0;i<arrPres.size();i++){
                    pres=(Presentacion)arrPres.get(i);
                    out.write("<option          value="+pres.getidPresentacion()+">"+pres.getpresentacion()
+"</option>)");
                    }
                  }
                %>
              </select>
            </td>
            <td></td>
          </tr>
          <tr class="Labels">
            <td></td>
            <td>Cliente: <input type="text" id="txtCliente" name="txtCliente"></td>
            <td></td>
            <td>Material:
              <select id="cboMaterial" name="cboMaterial">
                <%
                  Materiales mat = new Materiales();
                  ArrayList arrMat=DAOMateriales.arrMat();


                  if(arrMat !=null){
                    for(int i=0;i<arrMat.size();i++){
                    mat=(Materiales)arrMat.get(i);
                    out.write("<option value="+ mat.getidMaterial() + ">"+ mat.getmaterial() +"</option>)");
                    }
                  }
                %>
              </select>
            </td>
            <td> <td>Idioma:
              <select id="cboIdioma" name="cboIdioma">
                <option value="E">Español</option>
                <option value="I">Inglés</option>
              </select>
            </td></td>
            <td></td>
          </tr>
          <tr class="Labels">
            <td></td>
            <td>Notas Adicionales:</td>
            <td></td>
            <td colspan="4"><textarea name="txaNotas" id="txaNotas" rows="13" cols="45"></textarea>
          </tr>
          <tr>
```

```
            <td class="Labels"> </td>
          </tr>
          <tr>
            <td></td>
            <td><input type="submit" value="Enviar"></td>
          </tr>
        </table>
        </form>
        <%@include file="pie.jsp"%>
      </div>
  </body>
</html>
```

6.2.2.2 BuscaOrden

Al igual que la vista de AltaOrden, aquí se importan los paquetes que son indispensables para el funcionamiento correcto del mismo, aparece todo el código que se necesita para que el usuario pueda interactuar con el sistema mediante la pantalla BuscaOrden, el código de los campos y botones con que consta dicha pantalla.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.*"%>
<%@page import="com.diploweb.model.Orden"%>
<%@page import="com.diploweb.model.Productos"%>
<%@page import="com.diploweb.model.DAOOrden"%>
<%@page import="com.diploweb.model.DAOProductos"%>


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<%
  ArrayList arr = (ArrayList)request.getAttribute("arr");
  Orden ord = new Orden();
  Productos prod = new Productos();
%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sistema para Emisión de Órdenes de Empaque de Producto Terminado</title>
    <link href="<%=request.getContextPath() %>/vista/Estilos.css" type="text/css" rel="stylesheet" />
    <script src="<%=request.getContextPath() %>/vista/clsDatePicker.js"></script>
  </head>
  <script type="text/javascript">
    function fOcultaObjetos(tipoReporte){
      if (tipoReporte == '1')
      {
        document.getElementById('txtNoOrden').style.display = "block";
        document.getElementById('divInicio').style.display = "none";
        document.getElementById('divFin').style.display = "none";
        document.getElementById('divTipo').innerHTML = "Número de reporte:";
      }else{
        document.getElementById('txtNoOrden').style.display = "none";
        document.getElementById('divInicio').style.display = "block";
        document.getElementById('divFin').style.display = "block";
        document.getElementById('divTipo').innerHTML = "Entre las fechas:";
      }
    }
  </script>
  <body background="img/FONDO.PNG"leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
    <%@include file="menu.jsp"%>
     <div align="center">
```

```
<form name="Form1" method="post" action="<%=request.getContextPath() %>/EncuentraOrden">
    <table border="0" width="800" cellpadding="0" cellspacing="0">
      <tr class="tituloPrincipal">
        <td colspan="3">Búsqueda de Orden</td>
      </tr>
      <tr>
        <td></td>
        <td> </td>
      </tr>
      <tr>
        <td class="Labels">Búsqueda por:</td>
        <td class="Labels"><select name="cboBusqueda" id="cboBusqueda"
onchange="fOcultaObjetos(cboBusqueda.value);">
          <option value="1">Número de reporte</option>
          <option value="2">Fecha de reporte</option>
        </select></td>
      </tr>
      <tr>
        <td colspan="2">
        <table width="100%">
          <tr>
            <td class="Labels"><div id="divTipo">Número de reporte:</div></td>
            <td class="Labels">
              <input type="text" id="txtNoReporte" name="txtNoOrden" style="display: block;">
              <div id="divInicio" style="display: none;">
                <input type="text" id="txtFecInicio" name="txtFecInicio">
                <IMG alt="Calendario" src="<%=request.getContextPath()
%>/vista/img/calendar.gif"
                  border="0" class="Cursor_Hand"
onclick="show_calendar('Form1.txtFecInicio')">
              </div>
            </td>
            <td class="Labels">
              <div id="divFin" style="display: none;">
                <input type="text" id="txtFecFin" name="txtFecFin">
                <IMG alt="Calendario" src="<%=request.getContextPath()
%>/vista/img/calendar.gif"
                  border="0" class="Cursor_Hand" onclick="show_calendar('Form1.txtFecFin')">
              </div>
            </td>
          <tr>
        </table>
        </td>
      </tr>
      <tr>
        <td colspan="2" class="Labels" align="center"><input type="submit" value="Buscar"></td>
      </tr>
      <tr>
        <td></td>
        <td> </td>
      </tr>
    </table>
  </form>
  <%
  if (arr != null)
  {
  %>
  <table border="0" width="800" cellpadding="1" cellspacing="1">
    <tr class="tableTitulotd">
      <td>Fecha</td>
      <td>Número de órden</td>
      <td>Producto</td>
      <td>Lote</td>
      <td>Modificar</td>
      <td>Borrar</td>
    </tr>
  <%
```

```
            for(int i=0; i<arr.size(); i++)
            {
               ord = (Orden)arr.get(i);
               int idProducto = ord.getProducto();
               ArrayList arrProd = DAOProductos.arrProNom(idProducto);
               prod = (Productos)arrProd.get(0);
         %>
            <tr class="tabletd">
               <td><%=ord.getFecha()%></td>
               <td><%=ord.getOrden()%></td>
               <td><%=prod.getproducto()%></td>
               <td><%=ord.getLote()%></td>
               <td><a                                                    href="<%=request.getContextPath()
%>/vista/ModificaOrden.jsp?idOrden=<%=ord.getOrden()%>">
                  <img       src="<%=request.getContextPath()       %>/vista/img/edit.gif"       alt="Modificar"
border="0"></a>
               </td>
               <td><a                                                    href="<%=request.getContextPath()
%>/vista/borrar.jsp?idOrden=<%=ord.getOrden()%>">
                  <img       src="<%=request.getContextPath()       %>/vista/img/delete.gif"       alt="Borrar"
border="0"></a></td>
            </tr>
         <%
            }
         %>
            </table>
         <%
         }
         %>
      </div>
      <%@include file="pie.jsp"%>
   </body>
</html>
```

6.2.2.3 Error

Esta pantalla aparece en casos de que exista algún error al insertar la orden el código es muy simple, sólo consta de código HTML con el mensaje “Error al insertar”.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%--
The taglib directive below imports the JSTL library. If you uncomment it,
you must also add the JSTL library to the project. The Add Library... action
on Libraries node in Projects view can be used to add the JSTL 1.1 library.
--%>
<%--
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
```

```
    <h1>Error al insertar</h1>

    <%--
    This example uses JSTL, uncomment the taglib directive above.
    To test, display the page like this: index.jsp?sayHello=true&name=Murphy
    --%>
    <%--
    <c:if test="${param.sayHello}">
        <!-- Let's welcome the user ${param.name} -->
        Hello ${param.name}!
    </c:if>
    --%>

    </body>
</html>
```

6.2.2.4 ModificaOrden

Igual que en los archivos de BuscaOrden y AltaOrden, en este archivo se importaron algunos paquetes que se necesitan, además consta código HTML con las tablas, cuadros de texto y botones necesarios para poder modificar una orden.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="com.diploweb.model.Productos" %>
<%@page import="com.diploweb.model.Materiales" %>
<%@page import="com.diploweb.model.Presentacion" %>
<%@page import="com.diploweb.model.Orden" %>
<%@page import="com.diploweb.model.DAOProductos" %>
<%@page import="com.diploweb.model.DAOMateriales" %>
<%@page import="com.diploweb.model.DAOOrden" %>
<%@page import="com.diploweb.model.DAOPresentacion" %>
<%@page import="com.diploweb.model.DAOUbicacion" %>
<%@page import="java.util.ArrayList" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<%
    int idOrden = Integer.parseInt(request.getParameter("idOrden"));
    Orden ModOrden = new Orden();

    ArrayList arrModif = DAOOrden.buscaOrden(idOrden);
    ModOrden = (Orden)arrModif.get(0);
    String idioma = ModOrden.getIdioma();
    String espanol = "";
    String ingles = "";
    if (idioma == "E"){
        espanol = "selected";
    }else {
        ingles = "selected";
    }
%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link href="<%=request.getContextPath() %>/vista/Estilos.css" type="text/css" rel="stylesheet" />
         <script src="clsUtilerias.js" type="text/javascript"></script>
        <title>Sistema para Emisión de Órdenes de Empaque de Producto Terminado</title>
    </head>
    <body background="<%=request.getContextPath() %>/vista/img/FONDO.PNG">
        <div align="center">
            <%@include file="menu.jsp"%>
            <form method="POST" action="<%=request.getContextPath()%>/AddOrden">
```

```
        <table width="800px">
            <tr class="Labels">
                <td colspan="6"> </td>
            </tr>
            <tr class="Labels">
                <td>No. Orden:</td>
                <td><%=ModOrden.getOrden()%>
                </td>
                <td></td>
                <td>Producto:
                    <select id="cboProducto" name="cboProducto">
                        <%
                         int idProd;
                         Productos pro = new Productos();
                         ArrayList arrPro=DAOProductos.arrPro();
                         if(arrPro !=null){
                          for(int i=0;i<arrPro.size();i++){
                              pro=(Productos)arrPro.get(i);
                              idProd = pro.getidProducto();
                              if (idProd == ModOrden.getProducto()){
                                  out.write("<option value="+ pro.getidProducto() + " selected>"+ pro.getproducto()
+"</option>)");
                              }else{
                                  out.write("<option    value="+    pro.getidProducto()    +    ">"+    pro.getproducto()
+"</option>)");
                              }
                          }
                         }
                        %>
                    </select>
                </td>
                <td></td>
                <td>Fecha:</td>
                <td><%=ModOrden.getFecha()%></td>
            </tr>
            <tr class="Labels">
                <td></td>
                <td>No.Lote:          <input          type="text"          id="txtLote"          name="txtLote"
value="<%=ModOrden.getLote()%>">
                </td>
                <td></td>
                <td>Cantidad:        <input        type="text"        id="txtCantidad"        name="txtCantidad"
value="<%=ModOrden.getCantidad()%>">
                </td>
                <td></td>
                <td>Unidad de Medida:
                    <select id="cboUnidad" name="cboUnidad">
                        <%
                          int idPres;
                         Presentacion pres = new Presentacion();
                         ArrayList arrPres=DAOPresentacion.arrPres();
                         if(arrPres !=null){
                          for(int i=0;i<arrPres.size();i++){
                              pres=(Presentacion)arrPres.get(i);
                              idPres = pres.getidPresentacion();
                              if (idPres == ModOrden.getPresentacion()){
                                  out.write("<option    value="+    pres.getidPresentacion()    +    "    selected>"+
pres.getpresentacion() +"</option>)");
                              }else{
                                  out.write("<option        value="+        pres.getidPresentacion()        +        ">"+
pres.getpresentacion() +"</option>)");
                              }
                          }
                         }
                        %>
                    </select>
                </td>
```

```
                <td></td>
            </tr>
            <tr class="Labels">
                <td></td>
                <td>Cliente: <input type="text" id="txtCliente" name="txtCliente"
value="<%=ModOrden.getCliente()%>"></td>
                <td></td>
                <td>Material:
                    <select id="cboMaterial" name="cboMaterial">
                        <%
                            int idMat;
                            Materiales mat = new Materiales();
                            ArrayList arrMat=DAOMateriales.arrMat();
                            if(arrMat !=null){
                                for(int i=0;i<arrMat.size();i++){
                                    mat=(Materiales)arrMat.get(i);
                                    idMat = mat.getidMaterial();
                                    if (idMat == ModOrden.getMaterial()){
                                        out.write("<option value="+ mat.getidMaterial() + " selected>"+
mat.getmaterial() +"</option>)");
                                    }else{
                                        out.write("<option value="+ mat.getidMaterial() + ">"+ mat.getmaterial()
+"</option>)");
                                    }
                                }
                            }
                        %>
                    </select>
                </td>
                <td>Idioma:
                    <select id="cboIdioma" name="cboIdioma">
                        <option value="E" <%=espanol%>>Español</option>
                        <option value="I" <%=ingles%>>Ingles</option>
                    </select>
                </td></td>
                <td></td>
            </tr>
            <tr class="Labels">
                <td></td>
                <td>Notas Adicionales:</td>
                <td></td>
                <td colspan="4"><textarea name="txaNotas" id="txaNotas" rows="13"
cols="45"><%=ModOrden.getNotas()%></textarea>
            </tr>
            <tr>
                <td class="Labels"> </td>
            </tr>
            <tr>
                <td></td>
                <td><input type="submit" value="Enviar"></td>
            </tr>
        </table>
        </form>
        <%@include file="pie.jsp"%>
    </div>
  </body>
</html>
```

6.2.2.5 OrdenResultado

Esta pantalla sólo es para mostrar el resultado satisfactorio de la orden, el número de orden, la fecha y el número de lote.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.ArrayList"%>
<%@page import="java.text.*"%>
<%@page import="javax.net.*"%>
<%@page import="java.lang.*"%>
<%@page import="java.util.*"%>
<%@page import="java.io.*"%>
<%@page import="com.diploweb.model.Orden"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<%
   Orden ord = (Orden)request.getAttribute("orden");

   Date fec = new Date();
   SimpleDateFormat fch = new SimpleDateFormat("yyyyMMdd");
   String fecha = fch.format(fec);
%>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Sistema para Emisión de Órdenes de Empaque de Producto Terminado</title>
   </head>
   <body background="img/FONDO.PNG" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">
      <%@include file="menu.jsp"%>
       <div align="center">
       <table width="800px">
          <tr>
             <td class="subtitulo" colspan="2">El registro se agregó satisfactoriamente</td>
          </tr>
          <tr class="tabletd">
             <td>Número de Orden:</td>
             <td><%=ord.getOrden()%></td>
          </tr>
          <tr class="tabletd">
             <td>Fecha:</td>
             <td><%=fecha%></td>
          </tr>
          <tr class="tabletd">
             <td>Lote:</td>
             <td><%=ord.getLote()%></td>
          </tr>
       </table>
       </div>
       <%@include file="pie.jsp"%>
   </body>
</html>
```

6.2.3 Controlador

Como ya lo había mencionado antes, el controlador es el encargado de interactuar con el usuario con mensajes que provienen de la vista y decidir como la aplicación responderá a una acción es el que controla el flujo de datos.

6.2.3.1 AddOrden

Este archivo es el que actúa de modo interno para dar de alta una orden. En este archivo, primero que nada, se importan los paquetes necesarios para que este pueda funcionar, entre ellos se importa el diploweb.model.DAOOrden, y diploweb.model.Orden, que son los que generamos en el modelo y que contienen los atributos.

Después se mandan llamar a los atributos y se les asignan los parámetros que se le asignen y que sean correspondientes a cada uno de ellos que provengan de la vista, por ejemplo, al atributo producto se le asigna el valor que le envíe el usuario por medio de la vista InsertaOrden, desde el combo cboProducto. Estos datos se asignan a un objeto de tipo orden, los cuales se envían al método DAOOrden, donde, si la orden se insertó correctamente, se manda a llamar a la vista OrdenResultado que se explicó anteriormente, en caso de que no se inserte correctamente, se manda a llamar a la vista Error.

```
package com.diploweb.controller;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

import com.diploweb.model.DAOOrden;
import com.diploweb.model.Orden;
import java.text.*;
import java.util.*;

public class AddOrden extends HttpServlet {

   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

      int producto=Integer.parseInt(request.getParameter("cboProducto"));
      int lote=Integer.parseInt(request.getParameter("txtLote"));
      int cantidad=Integer.parseInt(request.getParameter("txtCantidad"));
      int unidad=Integer.parseInt(request.getParameter("cboUnidad"));
      String cliente=request.getParameter("txtCliente");
      int material=Integer.parseInt(request.getParameter("cboMaterial"));
      String notas=request.getParameter("txaNotas");
      String idioma=request.getParameter("cboIdioma");

      Date fec = new Date();
      SimpleDateFormat fch = new SimpleDateFormat("yyyyMMdd");
      String fecha = fch.format(fec);

      Orden o = new Orden();
```

```
        Orden ord = new Orden(0, producto, material, lote, cantidad, unidad, fecha, idioma, cliente, notas);

        o = DAOOrden.insertaOrden(ord);
        if(o != null){
            request.setAttribute("orden", o);
            request.getRequestDispatcher("/vista/OrdenResultado.jsp").forward(request, response);
        }else{
            request.getRequestDispatcher("/vista/Error.jsp").forward(request, response);
        }
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }

}
```

6.2.3.2 EncuentraOrden

En este archivo se recogen todos los parámetros introducidos desde la vista para llevar a cabo la búsqueda de una orden, primero se importan los paquetes necesarios como com.diploweb.model.orden, com.diploweb.DAOOrden, que son las clases que se generaron en el modelo.

Los valores obtenidos del formulario se almacenarán en un objeto de tipo ArrayList, que llevará el nombre arrBusqueda; los datos introducidos por el usuario en la vista BuscaOrden son enviados al método DAOOrden.

```
package com.diploweb.controller;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

import com.diploweb.model.Orden;
import com.diploweb.model.Utilidades;
import com.diploweb.model.DAOOrden;

public class EncuentraOrden extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        Utilidades oUt = new Utilidades();

        //Valores obtenidos de objetos del formulario
        int busqueda = Integer.parseInt(request.getParameter("cboBusqueda"));

        ArrayList arrBusqueda = new ArrayList();
```

```
        Orden obusca = new Orden();
        if (busqueda == 1){
            int idOrden = Integer.parseInt(request.getParameter("txtNoOrden"));
            arrBusqueda = DAOOrden.buscaOrden(idOrden);
        }
        else
        {
            String fecInicio = request.getParameter("txtFecInicio");
            String fecFin = request.getParameter("txtFecFin");
            fecInicio = oUt.formatFecha(fecInicio);
            fecFin = oUt.formatFecha(fecFin);
            arrBusqueda = DAOOrden.buscaOrden(fecInicio, fecFin);
        }
        request.setAttribute("arr", arrBusqueda);
        request.getRequestDispatcher("/vista/BuscaOrden.jsp").forward(request, response);

        out.close();
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit
the code.">
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

Conclusiones.

Un proyecto de desarrollo de un sistema comprende varios componentes o pasos que son llevados a cabo durante la etapa del análisis, el cual ayuda a traducir las necesidades del cliente en un modelo de Sistema que utiliza uno más de los componentes: Software, hardware, usuarios, bases de datos, documentación y procedimientos.

En una organización, el análisis y diseño de sistemas es un proceso en el cual se debe estudiar la situación con la finalidad de observar la manera de trabajar y así decidir si es necesario realizar una mejora, para esto el encargado de llevar a cabo dicha tarea es el analista de sistemas.

Antes de comenzar con el desarrollo de cualquier proyecto, es bueno realizar un estudio del sistema para detectar los detalles de la situación actual de la empresa. La información que se reúna con dicho estudio será de gran utilidad, como una base para crear estrategias de diseño.

En este trabajo se realizó un análisis y un diseño siguiendo un proceso el cual me fue de mucha utilidad, ya que conforme avanzaba en el análisis, me daba cuenta de los módulos que hacían falta o los que eran innecesarios de acuerdo a la forma en que lo había pensado.

En cuanto al lenguaje utilizado, Java fue un lenguaje diseñado orientado a objetos, los cuales agrupan en estructuras encapsuladas tanto sus datos, como los métodos que manipulan estos datos.

Una de las herramientas que me fueron de muchísima utilidad fue UML para llevar a cabo un mejor análisis del sistema, ya que una de las ventajas de dicha metodología, es que permite una gran comunicación entre los diversos actores acerca del modelo. UML me permitió darme cuenta de los procesos que me hacían falta y me ayudó bastante a aclarar algunas dudas y plasmar mejor la idea de cómo realizar el sistema.

Bibliografía

SCHMULLER, Joseph.
Aprendiendo UML en 24 horas.
Ed. Pearson Educación,
Primera edición, México 2000, p.p. 423.

E. FAIRLEY, Richard.
Ingeniería del Software.
Ed. Mc Graw-Hill,
Primera edición, 1987 p.p. 390.

GONZALO CUEVAS, Agustín.
Ingeniería del Software. Práctica de Programación.
Serie Paradigma. Ed. RA-MA,
Primera edición, E.U.A. 1991, p.p. 519.

J. BRAUDE, Erick.
Ingeniería del Software. Una perspectiva Orientada a Objetos.
Ed. Alfa Omega,
Primera Edición, 2003, p.p. 539.

CARBALLAR, José A.
Internet. Libro del navegante.
Ed. RA-MA,
Primera edición, Madrid 2000, p.p. 482.

FERREYRA C., Gonzalo.
Internet paso a paso. Hacia la autopista de la información.
Ed. Alfa Omega Gpo. Editor, S.A. de C.V.,
Primera Edición, p.p. 424.

PÉREZ, César,
MySQL para Windows y Linux.
Ed. Alfa Omega, Gpo. Editor, S.A. de C.V.
Primera Edición, México 2004, p.p. 454.

FOWLER, Martin y SCOTT, Kendall.
UML Gota a gota.
Ed. Addison Wesley Longman de México, S.A. de C.V.,
Primera Edición, 1999, p.p. 203.

Referencias de Internet

http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html

http://es.wikipedia.org/wiki/Ingenier%C3%ADa_del_software

http://www.inf.udec.cl/~ingsoft/software/isintroduccion.html

http://www.fceia.unr.edu.ar/asist/unidad13-4.pdf#search='herramientas%20de%20la%20ingenier%C3%ADa%20de%20Software'

http://www.javahispano.org/articles.print.action?id=76

http://redie.ens.uabc.mx/vol3no2/imprimir-contenido-mireles.html

http://neocygnus2.blogspot.com

http://java.ciberaula.com/articulo/tecnologia_orientada_objetos

http://www.inei.gob.pe/biblioineipub/bancopub/inf/lib5040/TECN.HTM

http://es.wikipedia.org/wiki/JSP

http://manuales.dgsca.unam.mx/webdina/servlets.htm

http://delta.cs.cinvestav.mx/~oolmedo/ClientServer/Servlets2.pdf#search='servlets'

http://www.htmlweb.net/redes/topologia/topologia_1.html

http://es.wikipedia.org/wiki/HTTP

http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/REDES02.htm