UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA GESTIÓN DE FALLAS Y CAPTURA DE REQUERIMIENTOS VÍA 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:
DANIEL UBALDO SÁNCHEZ NAVARRETE

ASESOR: MTRO. JESÚS HERNÁNDEZ CABRERA

MÉXICO **2007**

Capitulado.

Introducción. i

1 Fundamentos tecnológicos para el desarrollo de sistemas para Internet. 1
1.1 Introducción a Internet. 1
1.1.1 Concepto de Internet. 1
1.1.2 Historia de Internet. 1
1.1.3 Estructura de Internet. 2
1.1.4 Protocolo TCP/IP. 2
1.1.5 Direcciones IP. 3
1.1.6 FTP (File Transfer Protocol). 5
1.2 Redes. 6
1.2.1 Definición de Red. 6
1.2.2 Clasificación según su distribución lógica. 6
1.2.3 Clasificación según su tamaño. 8
1.3 Servicios de Internet. 9
1.3.1 Definición de Servicio de Internet. 9
1.3.2 Tipos de Servicios de Internet. 10
1.3.3 Protocolo http. 11
1.3.4 SSH . 11
1.4 Tecnología Orientada a Objetos. 12
1.4.1Introducción a la Tecnología Orientada a Objetos. 12
1.4.2 Historia de la Tecnología Orientada a Objetos. 12
1.4.3 Ventajas de la Programación Orientada a Objetos (POO). 13
1.4.4 Conceptos Básicos en la POO. 14
1.5 Java. 15
1.5.1 Descripción de Java. 15
1.5.2 Java Servlets 15
1.5.2.1 Ventajas de los Servlets. 16
1.5.3 JAVA JSP. 17
1.6 MySQL. 18
1.6.1 Historia de MySQL. 18
1.6.2 Características de MySQL 19
1.6.3 Administración 19

2 Ingeniería de software y UML. 20
2.1 La crisis del software. 20
2.1.1 Causas de la crisis del software. 20
2.1.2 Acontecimientos sobresalientes de la Crisis del Software. 21
2.1.3 Características de la Crisis del Software. 22
2.1.4 La Ingeniería del Software como solución a la crisis del software 23
2.2 Calidad del Software 24
2.2.1 Funciones y diseño de la Calidad del Software. 25
2.2.2 Métricas 25
2.2.3 Procesos de aseguramiento de calidad. 26
2.2.4 Técnicas de la caja negra y la caja blanca. 27
2.2.5 Estándar de IEEE para planes de aseguramiento de la calidad del software 29
2.3 Componentes de la Ingeniería del Software. 29
2.3.1 Procesos de la Ingeniería del Software. 29
2.3.2 Modelos del proceso del Software. 31
2.3.3 Herramientas del proceso de desarrollo. 34
2.3.3.1 Tecnología CASE y entornos de desarrollo. 34

2.4 UML. 36
2.4.1 Orígenes y beneficios del modelado con UML. 36
2.4.2 Diagrama de Casos de Uso. 38
2.4.3 Diagramas de Secuencia. 40
2.4.4 Diagramas de Colaboración. 40
2.4.5 Diagrama de Clases. 40
2.4.6 Diagrama de Estados. 43
2.4.7 Diagramas de Actividades. 44
2.4.8 Diagramas de Distribución. 45

3 Procesos de desarrollo en la ingeniería de software. 46
3.1 Definición de proceso de desarrollo. 46
3.1.1 RUP (Racional Unified Proccess). 46
3.1.2 XP (Extremme Programing). 48
3.1.3 PSP (Personal Software Process). 49
3.1.4 TSP (Team Software Process). 50

4 Análisis del sistema. 52
4.1 Enunciado general. 52
4.2 Problemática actual. 52
4.3 Requerimientos funcionales. 52
4.4 Requerimientos no funcionales. 53
4.5 Propuesta de solución. 53
4.6 Diagrama General de Casos de Uso. 54
4.6.1 Casos de Uso formato Expandido. 55
4.7 Diagramas de análisis. 65
4.7.1 Registrar Falla. 65
4.7.2 Registrar Requerimiento. 66
4.7.3 Insertar y Modificar Usuario. 66
4.7.4 Solucionar Falla. 67
4.7.5 Cotizar Requerimiento. 67

5 Diseño del Sistema. 68
5.1 Diagrama de Clases a Bajo Nivel. 68
5.2 Diagrama de Clases a Alto nivel. 69
5.3 ¿Qué es un Diagrama de Secuencia? 70
5.4 Diagramas de Secuencia del sistema. 71
5.4.1 Registrar Falla. 71
5.4.2 Registrar Requerimiento. 72
5.4.3 Consultar Solución. 73
5.4.4 Consultar Cotización. 74
5.4.5 Consultar Cotizar Requerimiento. 75
5.4.6 Consultar Solucionar Falla. 76
5.4.7 Insertar Usuario. 77
5.4.8 Actualizar Usuario. 78
5.4.9 Borrar Usuario. 79

6 Implementación. 80
6.1 Pantallas de Interfaz. 80
6.1.1 Pantalla de registro. 80
6.1.2 Pantalla Principal. 81
6.1.3 Registro de falla. 81
6.1.4 Registro de requerimiento. 82
6.1.5 Búsqueda de reportes. 83
6.1.6 Solución a falla. 84

6.1.7 Consulta solución a falla. 85
6.1.8 Cotización de requerimiento. 85
6.1.9 Consulta cotización de requerimiento. 86
6.1.10 Alta de usuarios. 86
6.1.11 Búsqueda de usuarios. 87
6.1.12 Actualiza usuario. 88
6.2 Documentación del desarrollo. 89
6.2.1 Modelo Vista Controlador (MVC del inglés Model View Controller). 89
6.2.2 Registro de Usuarios. 90
6.2.3 Registrar Falla. 96

Conclusiones 104
La Aplicación. 104
El Desarrollo y la Implementación. 105

Bibliografía. 106
Referencia en Internet. 107

I. Introducción.

En todo el mundo, el navegador Web se ha vuelto la forma dominante para acceder a contenidos, aplicaciones y sistemas. Ahora es común para las compañías dejar acceder a sus empleados, socios y clientes a una amplia gama de información y servicios a través de la Web.

El desarrollo de sitios Web se basa en el lenguaje HTML (HyperText Markup Languaje), todos los sitios Web contienen este lenguaje. Hasta los sitios web programados con otras tecnologías (como ASP, PHP, Java, etc.) que devuelven al navegador código HTML puro, para que este lo interprete y lo muestre al usuario.

La Web no sólo se limita a presentar textos y enlaces, sino que también puede brindar imágenes, videos, sonido y todo tipo de presentaciones, llegando a ser el servicio más rico en medios que tiene Internet. Por esta razón, al hablar del sistema que implementa el Web (hipertexto), se ha acuñado un nuevo término que es hipermedia, haciendo referencia a que la Web permite contenidos multimedia.

Actualmente se utilizan de forma dominante las aplicaciones Web en contraposición a los sitios Web estáticos. Todos, y supongo que usted también, desean que su presencia Web sea dinámica: presentar la última información, para permitir que los usuarios entren en el sistema y personalicen la apariencia del sitio, o para ofrecer la posibilidad de adquirir sus servicios on-line.

¿Cómo empezar?

Una buena forma de empezar a realizar sitios Web, es leer tutoriales y practicar.

También es aconsejable participar en los foros de discusión del tema en el que se tenga dudas, como puede ser el lenguaje HTML, o el uso de algún API, por ejemplo NetBeans, ya que de esta forma se obtendrán varias respuestas a una pregunta, con lo cual es posible obtener mejor información.

En el primer capítulo de este trabajo se explicarán los conceptos básicos que permitirán la introducción hacia el ambiente web e Internet, se hablará un poco de historia, así como de la estructura y la arquitectura de Internet, las redes y del desarrollo orientado a objetos.

Posteriormente, el capítulo dos, hace mención sobre la ingeniería de software, de sus características y acontecimientos además se mencionará brevemente los aspectos más importantes de la metodología UML.

El capítulo tres se tratará el tema acerca de los procesos de desarrollo en la ingeniería de software, como por ejemplo RUP, PSP o extreme programming.

Para el capítulo cuarto se mostrará el análisis del sistema mediante diagramas de caso de uso, (teoría explicada en el capítulo dos) y de análisis.

En el quinto capítulo, se encuentra el diseño del sistema, aquí se encuentran los diagramas de clases y los diagramas de secuencia que permitirán explicar a detalle la arquitectura de la aplicación.

Por último, en el capítulo sexto se muestra una ejemplificación de código de programación para el desarrollo del sistema, también se observarán figuras con las pantallas del prototipo.

A continuación se inicia con este trabajo.

1 Fundamentos tecnológicos para el desarrollo de sistemas para Internet.

Este capitulo se enfocará principalmente a explicar los fundamentos tecnológicos para el desarrollo de sistemas web, así como una breve reseña de conceptos e historia de lo que hoy conocemos como Internet.

Entre los conceptos básicos para la comprensión de lo que es Internet tenemos los tipos de protocolos de transferencia, las redes de datos y los servicios que se brindan a los usuarios. Se empezará por definir que es Internet.

1.1 Introducción a Internet.

Como sabemos Internet es el medio mas grande y poderoso de la actualidad para la transferencia de información y para entender mejor lo que es se menciona su concepto, historia y estructura.

1.1.1 Concepto de Internet.

Internet es un sistema mundial de redes de computadoras que está integrado por las diferentes redes de cada país del mundo, por medio del cual un usuario en cualquier computadora puede, en caso de contar con los permisos apropiados, acceder a la información de otra computadora y poder tener, inclusive, comunicación directa con otros usuarios en diferentes lugares.

Mediante Internet se transmite constantemente una enorme cantidad de información. Existen varios millones de personas que navegan por Internet en alrededor del mundo. Se dice navegar, ya que es normal encontrar información que proviene de distintas partes del mundo en una sola sesión.

Una de las ventajas de Internet es que es compatible la conexión con todo tipo de computadoras, desde las personales, hasta las más grandes que ocupan habitaciones enteras. Además actualmente nos facilita la vida, ya que, podemos realizar diferentes tipos de trámites por medio de esta red, tener comunicación con cualquier parte del mundo, incluso podemos ver conectados a la red cámaras de video, robots etcétera, y cada vez se va optimizando más, ahorrándonos tiempo y esfuerzo.

“Estar en Internet significa que una máquina opera con la pila de protocolos TCP/IP tienen una dirección IP y es capaz de enviar paquetes de IP a todas la máquinas de Internet”[1]

1.1.2 Historia de Internet.

Internet tiene una historia relativamente corta, pero fulgurante hasta el momento. Nació en EE.UU. a partir de un experimento realizado a principios de los años 70 por el Departamento de Defensa, un proyecto militar llamado ARPANET, con el objetivo de crear una red que permitiera comunicar a una

[1] Tanenbaum, Andrews. Redes de computadoras. Ed. Pearson. México 1997, p, 53

importante cantidad de computadoras de las instalaciones del ejército de EE.UU. y que pudiera seguir funcionando en caso de un desastre.

En 1985, la Nacional Science Foundation (NSF) creó NSFNET, una serie de redes informáticas dedicadas a la difusión de los nuevos descubrimientos y la educación. Basada en los protocolos de comunicación de ARPANET, la NSFNET creó un esqueleto de red nacional que fue ofrecido gratuitamente a cualquier institución americana de investigación o educación y fueron apareciendo otras redes regionales con el fin de poder enlazar el tráfico electrónico de instituciones individuales con el esqueleto de red nacional. Así se logró que creciera por todo el territorio de EE.UU.

La NSFNET fue creciendo junto con el descubrimiento por parte de los usuarios de su gran potencial y con la creación de nuevas aplicaciones que permitieran un más fácil acceso. Fue entonces cuando comenzó a extenderse Internet por otros países del mundo, abriendo un canal de comunicaciones entre Europa y EE.UU.

Internet crece a pasos agigantados mejorando los canales de comunicación con el fin de optimizar la rapidez de envío y recepción de datos. El Internet utiliza gran parte de los recursos existentes en las redes de telecomunicaciones.

1.1.3 Estructura de Internet.

Internet funciona con la estrategia "Cliente/Servidor", lo que significa que en la Red hay ordenadores Servidores que dan una información concreta en el momento que se solicite, y por otro lado están los ordenadores que piden dicha información, los llamados clientes.

Hoy en día, se han desarrollado redes que enlazaban computadoras de empresas o de particulares. Estas redes, eran de tipo LAN o WAN. Internet es una Red que está por encima de estas y que las une a todas. Lo que distingue al Internet es el uso del protocolo de comunicación llamado TCP/IP. Se ha establecido que en Internet, toda la información ha de ser transmitida mediante este protocolo.

1.1.4 Protocolo TCP/IP

TCP/IP (Transfer Control Protocol / Internet Protocol) es el protocolo[2] utilizado por todas las computadoras conectadas a Internet, de manera que estas puedan comunicarse entre sí. En Internet se encuentran conectadas computadoras de diferentes tipos, con hardware y software que podrían ser incompatibles. Una de las ventajas del TCP/IP, es que este protocolo se encarga de que la comunicación entre todos los tipos de computadoras se

[2] El Protocolo de Internet (IP, de sus siglas en inglés *Internet Protocol*) es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados. http://es.wikipedia.org/wiki/Protocolo_de_Internet

haga posible, ya que TCP/IP es compatible con todo tipo de sistema operativo y de hardware.

Esta arquitectura se empezó a desarrollar como base de ARPANET y con la expansión de INTERNET se ha convertido en una de las arquitecturas de redes más difundida.

Esta arquitectura TCP/IP se define por 4 niveles:

Nivel de subred. [Enlace y físico] Análogo al nivel físico del OSI (Open Systems Interconnection).

Nivel de interred. [Red, IP] Incluye al protocolo IP, que está encargado de enviar la información a su destino. Es utilizado con esta finalidad por los protocolos del nivel de transporte.

Protocolo proveedor de servicio [Transporte, TCP o UDP] Los protocolos de este nivel están encargados de llevar a cabo el manejo de datos con el objetivo de proporcionar la fiabilidad necesaria en el transporte de estos.

Nivel de aplicación. En este nivel están incluidos los protocolos responsables de proporcionar los servicios como correo electrónico, transferencia de ficheros (FTP), conexión remota y otros como http (Hypertext Transfer Protocol).
El protocolo IP es el principal del modelo OSI, así como parte integral del TCP/IP y sus principales funciones son el direccionamiento y la administración del proceso de fragmentación de los datagramas de información.

1.1.5 Direcciones IP

El datagrama[3], algunas veces identificada como datagrama Internet o datagrama IP, es la unidad de transferencia que utiliza el IP.

El protocolo IP no garantiza la entrega en secuencia, ya que se puede retrasar, enrutar (encaminar) incorrectamente o no completarse al dividir los fragmentos del mensaje. No está orientado a conexión, no tiene corrección de errores, ni control de congestión.

El ruteo puede darse paso a paso a todos los nodos, mediante tablas de rutas estáticas o dinámicas y direccionamiento IP.

El TCP/IP utiliza una dirección de 32 bits para identificar una computadora y la red a la cual se encuentra conectada. Únicamente el NIC (Centro de Información de Red) puede asignar las direcciones IP cuando la red está conectada a Internet, en caso de que la red no esté conectada a Internet, puede determinar su propio sistema de numeración.

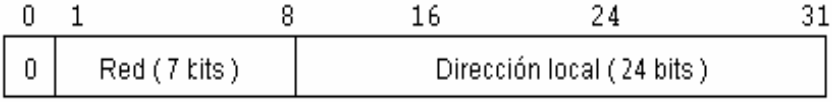
[3] Un datagrama es un fragmento de paquete que es enviado con la suficiente información como para que la red pueda simplemente encaminar el fragmento hacia el ordenador receptor, de manera independiente a los fragmentos restantes. http://es.wikipedia.org/wiki/Datagrama

Existen cuatro formatos para la dirección IP, desde la Clase A hasta Clase D (aunque últimamente se ha añadido la Clase E para un futuro) cada uno se utiliza dependiendo del tamaño de la red.

La clase es identificada por medio de las primeras secuencias de bits a partir de los tres primeros bits de orden más alto.

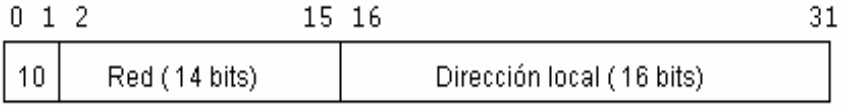
El formato de Clase A corresponde a redes grandes con varias computadoras. Las direcciones en decimal son 0.1.0.0 hasta la 126.0.0.0 (lo que permite hasta 1.6 millones de hosts).

CLASE A



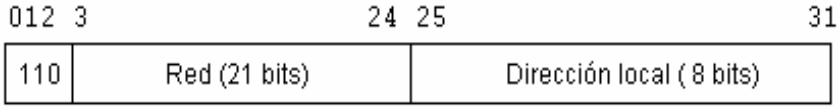
La Clase B es útil para redes que cuentan con un tamaño intermedio, su rango de direcciones varía desde el 128.0.0.0 hasta el 191.255.0.0. Esto permite tener 16320 redes con 65024 host en cada una.

CLASE B



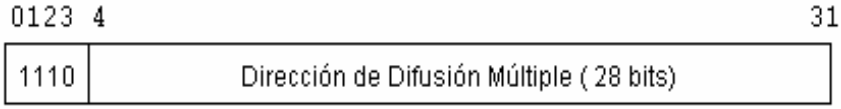
La Clase C tiene sólo 8 bits es útil para la dirección local (host) y 21 bits para red. Estas direcciones están comprendidas entre 192.0.1.0 y 223.255.255.0, lo cual permite cerca de 2 millones de redes con 254 hosts cada una.

CLASE C



Las direcciones de Clase D son utilizadas con el fin de la multidifusión, cuando se desea una difusión general a más de un dispositivo. El rango es desde 224.0.0.0 hasta 239.255.235.255.

CLASE D



Cabe mencionar que las direcciones de clase E en un futuro comprenderán el rango desde 240.0.0.0 hasta el 247.255.255.255.

Las direcciones IP son cuatro conjuntos de 8 bits, con un total de 32 bits. Estos bits se representan como si estuviesen separados por un punto, por lo que el formato de dirección IP puede ser red.local.local.local para Clase A hasta red.red.red.local para clase C.

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

En la Red no puede haber dos computadoras distintas con la misma dirección, ya que la información solicitada por alguna de las dos computadoras no sabría hacia cual de las dos dirigirse.

No es necesario que un usuario de Internet conozca las direcciones IP, ya que estas son manejadas por las computadoras por medio del protocolo TCP/IP invisiblemente, pero es necesario nombrar de alguna manera a las computadoras de Internet con el fin de definir a quién se pedirá la información y esto es mediante el Nombre de Dominio, que es prácticamente una traducción para los usuarios de la dirección IP.

Las computadoras que consultan Internet para pedir información no cuentan con un nombre de dominio, únicamente los servidores, que son las máquinas que reciben numerosas solicitudes de información.

1.1.6 FTP (File Transfer Protocol)

El FTP (File Transfer Protocol) es un sistema que permite transferir archivos y permite obtener o copiar archivos hacia una computadora remota de la red.

FTP transfiere archivos desde una máquina a otra; entre sus aplicaciones típicas nos permite el acceso a bases de datos públicas, y a archivos almacenados en computadoras centrales desde computadoras personales. También permite la distribución de información a través de la red.

FTP se basa en un conjunto de comandos que permiten el acceso a la información de una maquina remota.

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

“FTP tiene una especial importancia en la super-red, ya que es el protocolo que más se utiliza en la trasferencia de archivos entre computadoras conectadas en la red de Internet.”[4]

[4] Ferreyra, Gonzalo. Internet paso a paso. Hacia la autopista de la información. Ed. Alfa Omega. México D.F. 1997. p, 382

1.2 Redes

Una red es un sistema de transmisión de datos que permite el intercambio de información entre dos o más computadoras que se encuentran conectadas entre sí.

1.2.1 Definición de Red.

"Se ha definido como red a la conexión de varias computadoras a travéz de un cableado especial para compartir datos"[5]

Las redes se clasifican de acuerdo a su distribución lógica y su tamaño.

1.2.2 Clasificación según su distribución lógica.

Un servidor es una máquina que ofrece información a otras computadoras en Internet, existen servidores de páginas web, de base de datos, de correo, de impresión, de usuarios, etcétera.

Una máquina cliente es aquélla que solicita información a los servidores, por ejemplo, al consultar una página web o al solicitar un servicio de correo electrónico.

Una computadora puede ser servidor de un servicio pero cliente de otro servicio.

Existe un prototipo con el que deben cumplir todas las redes como el procesamiento de datos en grandes cantidades, dispositivos óptimos y confiabilidad al transportar los datos.

De acuerdo a dichos prototipos, el uso de las redes varía según la necesidad del cliente, por ejemplo:

- Compañías - centralizar datos.
- Compartir recursos (periféricos, archivos, etcétera).
- Confiabilidad (transporte de datos).
- Optimizar la disponibilidad de la información.
- Comunicación entre personal de las mismas áreas.
- Ahorro de dinero.

b) Topologías

Redes en Anillo. En esta topología, las estaciones se unen entre sí de manera que formen un círculo mediante un mismo cable. De esta forma, las señales circulan en un solo sentido alrededor del círculo y se regenera en cada nodo.

[5] Ferreyra, Gonzalo. Internet paso a paso. Hacia la autopista de la información. Ed. Alfa Omega. México D.F. 1997. p, 31

Una de las ventajas en esta topología es que los cuellos de botella son poco frecuentes.

La desventaja es que como existe sólo un canal de comunicación para todas las redes, en caso de que este falle o alguna de las estaciones, todas quedarán incomunicadas. Este problema se puede resolver si se pone un canal alterno en caso de que el otro falle.

Para la resolución de problemas en la transmisión de datos existe un mecanismo:

Token Ring: En este mecanismo la estación es conectada al anillo por la unidad de interfaz (RIU), cada RIU se responsabiliza de llevar el control cuando pasan los datos por ella, de la regeneración de la transmisión y el paso a la siguiente estación. En caso de que la dirección de cabecera de una transmisión indique que los datos son para una estación en concreto, la unidad de interfaz los copia y pasa la información a la estación de trabajo conectada a la misma.

Este mecanismo es usado en redes de area local, el token va de estación en estación en forma cíclica, inicialmente en estado desocupado. Cuando una estación tiene el token y quiere realizar una transmisión, cambia su estado a ocupado, agregando los datos atrás y poniéndolos en la red. En caso de que no haga transmisión, pasa el token a la estación siguiente. Cuando el token regresa a la estación que ya transmitió, saca los datos y lo pone en desocupado regresándolo a la red.

Bus: permite que todas las estaciones reciban la información que se transmite, una estación trasmite y todas las restantes escuchan.

Las ventajas que tiene esta topología son que si surge una falla en una estación no afectará al resto de la red, además, necesita una cantidad menor de cables.

Las desventajas de esta topología son que cuando hay sólo un canal que comunica a las estaciones de red, en caso de que falle el canal, las restantes quedan incomunicadas; el problema se puede resolver poniendo un bus paralelo para estos casos o utilizando algoritmos para el aislamiento de componentes defectuosos.

Para la resolución de problemas en la transmisión de datos existen dos mecanismos:

CSMA/CD: Todas las estaciones compiten por el uso del canal, ya que se les considera a todas iguales, estas redes son con escucha de colisiones, cada vez que alguna estación desea hacer una transmisión de datos debe escuchar el canal, en caso de que ya se esté haciendo una transmisión por alguien más, debe esperar a que termine, y en caso de que no se esté efectuando transmisión, esta estación transmite y se queda escuchando posibles choques, en caso de que así sea debe esperar un tiempo y volver a intentarlo.

Token Bus: Este mecanismo resuelve el problema de colisiones del mecanismo anterior. Se utiliza un token (trama de datos) que pasa de estación en estación en forma cíclica, es decir forma un anillo lógico. Cuando una estación tiene el token, tiene el derecho exclusivo del bus para transmitir o recibir datos por un tiempo determinado y luego pasa el token a otra estación, previamente designada y las demás estaciones no pueden hacer ninguna transmisión si no tienen el token, sólo escuchar y esperar turno.

Red en Estrella: En esta topología la red se une en un único punto, tiene un control centralizado como un concentrador de cableado.

Red Bus en Estrella: En esta topología la red es un bus con un cableado físico como estrella mediante concentradores y tiene el objetivo de facilitar la administración de la red.

Red en Estrella Jerárquica: Esta red funciona mediante concentradores en cascada par formar una red jerárquica y es utilizada en la mayoría de las redes locales.

1.2.3 Clasificación según su tamaño.

Redes PAN (red de administración personal). Son pequeñas redes que se conforman por máximo 8 máquinas, un ejemplo de eso es un café Internet.

Las redes LAN (Local Area Network). Son redes de área local y son utilizadas en una empresa, son pequeñas y aptas para una oficina. Tienen dimensiones limitadas y restringidas en tamaño, por lo cual son rápidas y las estaciones se pueden comunicar entre sí y su administración está simplificada.

Las redes LAN utilizan tecnología de difusión por medio de un cable coaxial al que se conectan todas las máquinas y su velocidad es entre 10 y 100 Mbps.

Las características de las redes LAN son:

Los canales son propiedad de los usuarios, la tasa de error es menor que las redes WAN, las estaciones están cerca entre sí, aumenta la productividad y eficiencia en los trabajos de oficina al compartir la información, los enlaces son de alta velocidad. Las rede LAN utilizan tecnología de transmisión que se da por un cable donde todas las computadoras se conectan.

CAN (Campus Area Network, Red de Area Campus). Es una colección de redes LAN que pertenecen a una sola entidad en un área delimitada en kilómetros y que utiliza tecnologías como Gigabit Ethernet con el objetivo de lograr una conectividad mediante fibra óptica y se dispersan dentro de una empresa o un campus universitario, etcétera.

Las redes WAN (Wide Area Network, redes de área extensa) Este tipo de redes comunican países y continentes, debido a esto, sus velocidades son menores que en las redes LAN, sin embargo, tienen la capacidad de transferir una mayor cantidad de datos y conectar una gran cantidad de computadoras

llamadas hosts. Las líneas que utiliza este tipo de redes pueden ser parte de redes públicas de transmisión de datos.

Normalmente, este tipo de redes son conectadas a redes WAN, con la finalidad de proporcionar acceso a mejores servicios como Internet. Las redes WAN son mucho más complejas, porque deben enrutar correctamente toda la información proveniente de las redes conectadas a esta.

Una subred se forma por dos componentes:

- Las 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 de información pase de un router a otro, debe pasar por routers intermedios, cada uno recibe el paquete por una línea de entrada almacenándolo y una vez que se libera la línea de salida, retransmite el paquete de información.

Redes MAN (Metropolitan Area Network, redes de área metropolitana). Su ubicación geográfica se determina por ciudad o municipio, su cobertura es mayor a 4 Km. Este tipo de redes tienen un par de buses con una transferencia de datos independiente uno de otro con una sola dirección. El mecanismo para la resolución de problemas en la transmisión de datos que usan las redes MAN, es DQBD, que consiste en donde hay dos buses unidireccionales en los que están conectadas las estaciones y cada uno tiene su cabecera y su fin. En caso de que una computadora desee trasmitir a otra y su ubicación es a la izquierda, utiliza el bus de arriba, de lo contrario usará el de abajo.

Redes Punto a Punto. En estas redes, la computadora tiene la capacidad de actuar como cliente y servidor, también facilitan que se compartan los datos y periféricos con facilidad para un grupo pequeño de usuarios, aunque la seguridad es un poco complicada debido a que la administración no está centralizada.

Redes Basadas en servidor. Estas redes son mejores para compartir gran cantidad de recursos y datos, pueden tener más de un servidor dependiendo del volumen del tráfico, número de periféricos, etcétera. Un administrador puede supervisar la operación de la red y la seguridad.

1.3 Servicios de Internet.

Un servicio de Internet es una aplicación que se le proporciona al usuario para que pueda hacer uso de él, por ejemplo, el correo electrónico que brindan algunos sitios de la web son un tipo de servicio web. A continuación se define un servicio en Internet.

1.3.1 Definición de Servicio de Internet.

Los servicios de Internet son todas las posibilidades para satisfacer necesidades del usuario que ofrece Internet. Cada servicio es una manera de

sacarle provecho a Internet. Un usuario puede especializarse en el manejo de sólo uno de estos servicios sin necesidad de saber nada de los otros, aunque es muy importante que conozca todos o la mayoría con el fin de darle una buena utilidad a Internet.

1.3.2 Tipos de Servicios de Internet.

Actualmente, los servicios más conocidos y utilizados en Internet son: Correo Electrónico, World Wide Web, FTP, Grupos de Noticias, IRC y Servicios de Telefonía.

Correo Electrónico: El correo electrónico (e-mail) es el servicio de envío y recepción de mensajes de texto, es un medio rápido, eficiente y sencillo de utilizar.

El correo fue una de las primeras aplicaciones que fueron creadas para Internet. Para muchos usuarios, ha reemplazado prácticamente al servicio postal para breves mensajes por escrito, ya que es más rápido y barato enviar un e-mail que una carta postal. También pueden efectuarse conversaciones en vivo con otros usuarios en otras partes del mundo utilizando el IRC (Internet Relay Chat). Más recientemente, el software y hardware para telefonía en Internet permite conversaciones de voz en línea.

Si el servidor llega a fallar, los mensajes enviados no se pierden, lo que hacen es que retienen el último punto hasta que pueda restablecerse un modo de llegar a su destino.

Web: La World Wide Web (WWW), es un servidor de información. Es un sistema de distribución de información tipo revista. En Internet se almacenan las páginas Web, que son páginas de texto con gráficos o fotos. A partir de la invención de la www, muchas personas empezaron a conectarse a la Red desde sus domicilios, como entretenimiento. Internet recibió un gran impulso, hasta el punto de que hoy en día, casi siempre que hablamos de Internet, nos referimos a la WWW.

World Wide Web, es el universo de información accesible a través de Internet, es el componente más usado en Internet. Su característica más importante es el texto remarcado para referencias cruzadas instantáneas. En la mayoría de los Sitios Web, ciertas palabras aparecen en texto de otro color diferente al resto del documento. Generalmente, este texto es subrayado. Al seleccionar una palabra o frase, uno es transferido al sitio o página relacionada a esa frase. En algunas ocasiones hay botones, imágenes, o porciones de imágenes que pueden activarse mediante un clic.

El Web proporciona acceso a millones de páginas de información. La apariencia de un Sitio Web puede variar ligeramente dependiendo del explorador que use. Las versiones más recientes proporcionan una funcionalidad más óptima cada vez.

Web Marketing: Es la mercadotecnia en Internet. En este, un sitio Web y sus productos o servicios llegan al mercado, empleando herramientas que Internet ofrece para realizar la promoción del sitio Web.

Grupos de Noticias: Son lugares en Internet en los cuales la gente se reúne para entablar debates sobre temas técnicos. Existen miles de grupos virtuales de noticias cubriendo temas desde entretenimiento, noticias, asuntos sociales, literatura y ciencia, negocios, salud, política, etcétera.

Servicio IRC (Internet Relay Chat)**.** Este servicio nos permite entablar conversaciones con una o varias personas mediante mensajes de texto en un tiempo real. Permite también el envío de imágenes u otro tipo de ficheros mientras se está charlando.

1.3.3 Protocolo HTTP

El protocolo HTTP (HyperText Transfer Protocol) es un protocolo de transferencia que se usa en cada transacción de la web donde el contenido de las páginas web y el protocolo de transferencia es el sistema mediante el cual se envían las solicitudes cuando se desea acceder a una página web y regresando la información como pueden ser los formularios.

HTTP es un protocolo que es incapaz de guardar información acerca de conexiones pasadas. Cuando se termina de realizar la transacción se pierden todos los datos, motivo por el cual se hicieron más frecuentes de utilizar las cookies, que son ficheros guardados en la computadora y que son capaces de leer el sitio web cuando se establece la conexión con él, de este modo reconocen a un visitante que ya estuvo anteriormente en ese sitio. Este protocolo está basado en el modelo cliente-servidor, en el que un cliente HTTP inicia una conexión en la cual envía una solicitud al servidor, el cual responde enviando el recurso solicitado y así se cierra la conexión.

1.3.4 SSH

SSH (Secure SHell) es un software que tiene la función de permitir una conexión segura a sistemas mediante canales inseguros, también es usado para ejecutar órdenes en ese sistema remoto o transferir ficheros desde o hacia él, de un modo fiable.

SSH proporciona los siguientes tipos de protección:

El cliente tiene la posibilidad de verificar que está efectuando la conexión al mismo servidor que otras ocasiones.

Los datos enviados y recibidos durante la conexión son transferidos mediante una fuerte encriptación para dificultar la lectura.

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).

SSH facilita realmente el hecho de cifrar tipos diferentes de comunicación que normalmente se envía en modo inseguro a través de redes públicas.

1.4 Tecnología Orientada a Objetos.

La Tecnología Orientada a Objetos es un nuevo enfoque acerca de la forma de organizar las diferentes piezas que integran un sistema de información, como en el hardware.

1.4.1Introducción a la Tecnología Orientada a Objetos.

Actualmente, la tecnología orientada a objetos además de aplicarse a los lenguajes de programación, también se aplica en el análisis y diseño con gran éxito y a las bases de datos, ya que para hacer una buena programación orientada a objetos es necesario desarrollar el sistema entero aplicando este tipo de tecnología.

La programación orientada a objetos ha sido bien admitida en el desarrollo de proyectos de software desde los últimos años. Esta bienvenida se debe a que tiene grandes ventajas y capacidades que las antiguas formas de programar no tenían.

1.4.2 Historia de la Tecnología Orientada a Objetos.

La tecnología orientada a Objetos surge en los años 60, cuando se tuvo la necesidad de describir y simular fenómenos como sistemas de comunicación, redes neuronales, sistemas administrativos, etcétera.

En 1961 Krystin Nygaard tuvo la idea de desarrollar un lenguaje de doble propósito (descripción de sistema y simulación programable) y crea SIMULA I. Los usuarios descubrieron que también proveía de nuevas facilidades cuando se utilizaba para otros propósitos como el prototipo y las aplicaciones.

En 1967 se creó SIMULA 67, y en él se implementaron por primera vez los conceptos de clase, objeto y herencia, que serían elementos importantes en los Lenguajes Orientados A Objetos.

En 1970 se crea en la Corporación Xerox el Smalltalk, que fue el primer lenguaje exclusivamente orientado a objetos y fue importante por sus herramientas de desarrollo y por su lenguaje.

En la década de los 80 Smalltalk evoluciona y se crea ADA. En estos lenguajes es muy importante la abstracción de datos. Los problemas del mundo real son representados por medio de objetos a los cuales se le agrega operaciones cuando se necesita.

Tradicionalmente, la programación fue hecha de un modo secuencial o lineal, estaba hecha por una serie de pasos consecutivos con estructuras consecutivas y bifurcaciones.

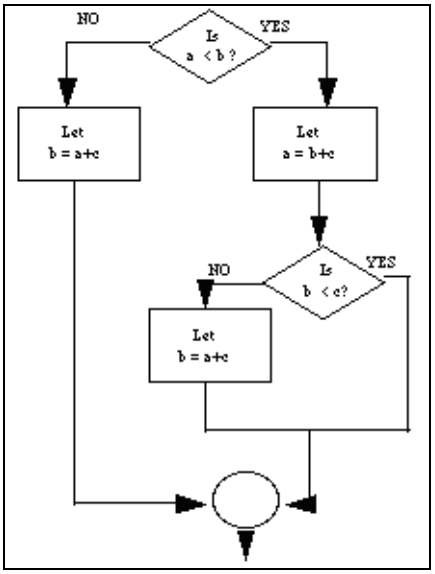


Fig. 1.1 Diagrama de flujo de la programación secuencial.

Cuando los lenguajes se basaban en este modo de programación ofrecían ventaja cuando los sistemas tienen un grado de complejidad bajo, pero cuando son más complejos estos programas no son flexibles y se vuelve muy complicado mantener líneas de código en grandes cantidades en un solo bloque. Debido a este tipo de problemas, han ido apareciendo lenguajes que se basan en la programación estructurada, la cual, tiene como objetivo principal separar el programa en módulos de las partes complejas, de modo que sean ejecutados según se vayan requiriendo y así tener un diseño compuesto por módulos independientes comunicados entre sí.

De esta manera, la evolución que se fue dando en cuanto a la programación siempre descomponía el programa poco a poco, lo cual condujo directamente a la programación orientada a objetos, que permite a los desarrolladores crear sistemas de nivel empresarial y con reglas de negocios más complejas. La Programación Orientada a Objetos (POO) surge de una obligada evolución de la programación estructurada que simplifica la programación con los nuevos conceptos que tiene y con los que trabaja y que está basada en dividir el programa en pequeñas unidades lógicas de código a las que se les llama objetos.

1.4.3 Ventajas de la Programación Orientada a Objetos (POO)

La POO proporciona conceptos y herramientas con los que se puede modelar y representar el mundo real lo más fielmente posible.

También fomenta la reutilización y extensión del código, facilita la creación de programas visuales, facilita el mantenimiento el software, permite crear

sistemas más complejos y la construcción de prototipos, facilita el trabajo en equipo y agiliza el desarrollo de software.

En la POO se considera un programa como un sistema de objetos que interactúan entre sí donde el ambiente de desarrollo facilita la construcción de programas y solución de problemas, ya que permite al desarrollador abstraer la interfaz gráfica del problema del código normal de un programa. Los problemas son considerados y se resuelven con módulos de código gigante denominados clase y que contienen el código necesario como las variables, funciones, interfaces, procedimientos, etcétera, que permiten la solución del problema.

1.4.4 Conceptos Básicos en la POO

Algunos de los conceptos básicos para lograr la comprensión de la POO son los siguientes.

Objeto: Un objeto es la instanciación de una clase, Es una cosa tangible, o que se puede comprender intelectualmente, tiene un estado y un funcionamiento. El estado está contenido en sus variables y el funcionamiento se determina por sus métodos. El funcionamiento del objeto es activado cuando se invoca a uno de sus métodos, que realizará una acción o modificará su estado.

Clases: Una clase es la definición de tipo de dato y está formada por un nombre, atributos y métodos. Una clase es un prototipo que define las variables y los métodos comunes a un objeto. De las clases se pueden crear varios objetos del mismo tipo. Cada objeto tendrá sus propios valores y compartirán las mismas funciones y antes de crear un objeto se debe crear la clase.

Atributo: Los atributos de una clase definen sus características y tienen algún valor. Los atributos pueden ser de diferentes tipos de datos.

Método: Un método define el comportamiento de la clase, establece las interfaces de comunicación con el código, puede o no recibir parámetros de entrada y retorna un tipo de datos.

Mensajes: Para poder crear una aplicación se necesita más de un objeto, y los objetos no se pueden aislar unos de otros, para comunicarse esos objetos se envían mensajes, que son simples llamadas a las funciones o métodos del objeto con el se quiere comunicar para decirle que haga cualquier cosa.

Herencia: Una herencia es cuando se crea una clase a través de una clase existente, y esta clase tendrá todas las variables y los métodos de su superclase; además se le podrán añadir otras variables y métodos propios, es decir, hereda las propiedades de la superclase.

Polimorfismo: Permite múltiples implementaciones de métodos, dependiendo del tipo de objeto que se indica al invocar el método correspondiente. De esta manera el polimorfismo permite que el mismo mensaje enviado a diferentes objetos resulte en acciones dependientes del objeto que recibe el mensaje.

1.5 Java.

Hoy en día existen muchos lenguajes y plataformas para el desarrollo web, entre ellas tenemos al Visual Studio Net de Microsoft que ofrece gran variedad de lenguajes, algunos otros lenguajes son: CGI, PHP, y por supuesto Java, esta última es una de las mas robustas que existen en el mercado y tiene la ventaja de ser software libre. En seguida se brinda una breve descripción de este lenguaje.

1.5.1 Descripción de Java

Java es un lenguaje de programación con el cual se puede realizar cualquier tipo de programa. Actualmente, Java es un lenguaje muy extendido que cada vez tiene mayor importancia en el ámbito de Internet como en Informática en general.

Una de las ventajas de Java es que es un lenguaje muy independiente de la plataforma, ya que se ha creado una máquina de Java para cada sistema que sirve como puente entre el sistema operativo y el programa de java posibilitando así el funcionamiento en cualquier computadora independientemente del sistema operativo, ya sea Windows, Linux, Apple, etcétera, por este motivo es tan interesante para Internet, ya que a esta red acceden muchas personas desde diferentes computadoras. Últimamente Java se ha ido desarrollando para distintos dispositivos como agendas, móviles y para cualquier necesidad que surja para la industria.

Uno de los primeros triunfos de Java fue que se integró en el navegador Netscape y permitía ejecutar programas dentro de una página Web, hasta entonces impensable con el HTML[6]. Java es un lenguaje potente, seguro, universal y puede ser utilizado por todo el mundo, ya que es gratuito.

Actualmente existe un amplio abanico de posibilidades, casi todo lo que puede programarse con cualquier lenguaje, puede programarse en Java con mayores ventajas.

Java implementa la tecnología básica de C++ con funcionalidades inexistentes como la resolución dinámica de métodos y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje, trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta características propias de la orientación a objetos que son: herencia, polimorfismo y encapsulamiento.

1.5.2 Java Servlets

Los Applets son programas que pueden cargarse a través de una red y que se ejecutan igual en cualquier plataforma debido a las características de Java, que

[6] El HTML fue el lenguaje que se creó para compartir documentos en la Web. En el pasado los recursos eran limitados, así que tanto el protocolo HTTP como el lenguaje HTML tenían que ser muy sencillos. http://html.conclase.net/tutorial/html/1/3

se utiliza principalmente para proveer a las páginas web de mayor interactividad por medio de los Applets, por ello sólo se actuaba del lado del cliente. Pero actualmente el servidor también se ve beneficiado de las ventajas que ofrece Java gracias a los Servlets.

Los Servlets son programas que funcionan como los CGIs convencionales, atendiendo peticiones de un cliente y teniendo como encargado al servidor pero escritos en Java y con la ventaja de explotar todas sus propiedades.

Por ejemplo, un Servlet puede ser responsable de tomar los datos de un formulario HTML y enviarlos a una base de datos para actualización de la misma.

Anteriormente los CGIs eran los únicos que tenían interacción entre el cliente y el servidor. Los CGIs son los típicos formularios que el usuario llena con sus datos que posteriormente forman parte de una base de datos.

Los Servlets se ejecutan en el servidor y no presentan ningún tipo de interfaz gráfica, ya que se encargan de hacer el trabajo oculto.

Los Servlets pueden sustituir a los CGIs, ya que proveen la manera de generar documentos dinámicos fáciles de escribir y ejecutar, además evitan problemas de desarrollar el programa según la plataforma utilizada, ya que los Servlets son una extensión estándar de Java. Se pueden cargar de forma transparente tanto desde un disco local como de una dirección remota.

Además los Servlets pueden comunicarse entre sí y es posible la reasignación dinámica de la carga de proceso entre diversas máquinas. Pueden reutilizarse CGIs ya elaborados e incrustarlos en Servlets.

1.5.2.1 Ventajas de los Servlets.

Desempeño: Los Servlets son más rápidos que los CGI.

A diferencia de los CGI, los Servlets requieren que se cuente con una máquina virtual de Java (JVM) corriendo sobre el servidor todo el tiempo. Esta máquina permite usar menos los recursos del sistema e incrementar el desempeño cuando los sitios están ocupados.

Portabilidad: Los Servlets son tan portables como cualquier otra aplicación de Java. La portabilidad con los servlets de Java es simple, debido a que Java fue diseñado para ser portable a través de todas las plataformas, permitiendo que las aplicaciones sean movidas fácilmente de un sistema operativo a otro.

Seguridad: Los lenguajes compilados como Java proveen mejor seguridad que los lenguajes que interpretan Scripts.

Los Servlets son archivos de clases compilados mientras que un CGI/Perl es manipulado en su forma de código fuente. Dependiendo quien tenga acceso al servidor Web, se puede elegir entre instalar o no el código fuente.

Arquitectura de los Servlets.

La interfaz Servlet está provista de métodos que manipulan a los servlets y la comunicación con sus clientes.

Cuando un Servlet es llamado desde un cliente, este recibe dos objetos: ServletRequest y ServletResponse. La interfaz ServletRequest es la encargada de mantener la comunicación desde el cliente al servidor, la interfaz ServletResponse atiende la comunicación desde Servlet al cliente.

La interfaz ServletRequest permite el acceso a la información al Servlet como, los nombres de parámetros pasados por el cliente, el protocolo usado por el cliente, y los nombres de los host remoto que hacen la solicitud y el servidor que la recibe. Esta interfaz permite a los Servlets el acceso a métodos que permiten manejar la presentación de la respuesta como salida en el navegador, a través de los cuales consiguen los datos desde el cliente que usa protocolos como http post, etc.

La interfaz ServletResponse proporciona al Servlet los métodos para contestarle al cliente. Permite al Servlet configurar la forma de salida de los datos para el cliente, ServletOutputStream permite enviar la replica de datos como respuesta. Las subclases de ServletResponse le dan más capacidad de respuesta al Servlet.

Las clases e interfaces descritas conforman a un servlet básico. Pero existen métodos adicionales que provee la API con la capacidad para controlar sesiones o múltiples conexiones, entre muchas más aplicaciones

1.5.3 JAVA JSP.

JavaServer Pages (JSP) es la tecnología para generar páginas web de una manera dinámica en el servidor, desarrollado por Sun Microsystems, basado en Scripts que utilizan una variante del lenguaje Java.

JavaServer Pages, es una tecnología Java que permite a los programadores generar dinámicamente HTML, XML[7] o algún otro tipo de página web. En las JSP, se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente. El código java se incluye dentro de las marcas de etiqueta <% y %>.

La ventaja de JSP frente a otros lenguajes es que permite integrarse con clases Java, lo que permite separar en niveles las aplicaciones web, almacenando en clases Java las partes que consumen más recursos así como

[7] XML, es el estándar de Extensible Markup Language y no es más que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes.
http://geneura.ugr.es/~maribel/xml/introduccion/index.shtml#12

las que requieren más seguridad, y dejando la parte encargada de formatear el documento html en el archivo JSP.

Java se caracteriza por ser un lenguaje que puede ejecutarse en cualquier sistema, lo que sumado a JSP le da mucha versatilidad. Es otra de las nuevas tecnologías para tratar de hacer más eficiente el modelo cliente-servidor y sobre todo la construcción de sistemas de comercio electrónico.

En este modelo una pagina html también incluye código en Java, es el servidor de páginas quien al estar mandando la página a la computadora remota, la compila y la convierte en un Servlet.

Esta tecnología combina en una sola aplicación, código html y código Java.

El proceso de crear un JSP es sencillo, primero se crea un achivo normal con notepad combinando código html y código java, se graba con extensión ****.jsp, se hace un FTP al servidor y listo. Cuando el usuario requiere un JSP el servidor lo carga, lo compila, lo convierte a Servlet y manda la página resultante al usuario remoto.

1.6 MySQL

MySQL es un Sistema de administración de Base de Datos SQL y es una implementación Cliente-Servidor que tiene un servidor y diferentes programas o librerías que actúan como clientes. Mediante MySQL se pueden procesar datos previamente grabados en una base de datos y agregar o acceder a los mismos.

MySQL es un software de código abierto, ya que es accesible para que cualquiera pueda usarlo o modificarlo y para usarse en volúmenes de datos tanto grandes como pequeños. Es confiable, rápido y seguro, lo que hace que sea altamente conveniente para lograr el acceso a base de datos en Internet.

Se puede descargar MySQL desde Internet gratuitamente, ya que utiliza el GPL (GNU Licencia Publica General), que nos define lo que se puede o no hacer con el software, de modo que cualquiera puede estudiar el código fuente y modificarlo para adaptarlo a sus necesidades.

1.6.1 Historia de MySQL.

En 1981 IBM comenzó a comercializar el SQL, que ha tenido un importante lugar en el desarrollo de bases de datos relacionales.

En 1983 surgió DB2, que es la más popular de las bases de datos al menos en las grandes máquinas.

En la década del 90, Michael Windenis comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo nivel. Sin embargo, llegó a la conclusión que mSQL no era tan flexible ni rápido para cubrir sus necesidades. Fue entonces cuando surgió en una nueva interfaz para lograr portar aplicaciones y utilidades de MiniSQL a MySQL con una mayor facilidad.

1.6.2 Características de MySQL

- El principal objetivo de MySQL es velocidad y robustez.
- Multiproceso, puede usar varias computadoras si estas están disponibles.
- Tiene la capacidad de trabajar en distintas plataformas y sistemas operativos.
- Tiene un flexible y seguro sistema de privilegios y contraseñas.
- Todas las palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- Todas las columnas pueden tener valores por defecto.
- Es de gran utilidad para chequear, 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.
- Tiene 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.

Lo primero que se debe hacer es arrancar el servidor MySQL para realizar cualquier operación con la base de datos.

1.6.3 Administración

El sistema de seguridad de MySQL garantiza que cada usuario pueda hacer las cosas que le están permitidas.

El sistema decide los diferentes privilegios dependiendo del usuario, de la base de datos y de la computadora a que se conecte. El sistema de privilegios se basa en el contenido de 5 tablas de la base de datos de MySQL que son: host, user, db, tables_priv, y colums_priv.

La tabla user contiene información sobre los usuarios, desde qué máquinas se puede acceder al servidor MySQL, su clave y de sus diferentes permisos. La tabla host informa sobre qué máquinas podrán acceder a nuestro sistema, así como a las bases de datos que tendrán acceso y sus diferentes permisos. Finalmente, las tablas db, tables_priv, columns_priv nos proveen de un control individual de las bases de datos, tablas y columnas.

2. Ingeniería de software y UML.

"La ingeniería del software es el proceso de construir aplicaciones de tamaño o alcance prácticos, en las que predomina el esfuerzo del sofware y que satisfacen los requerimientos de funcionalidad y desempeño" [8]

2.1 La crisis del software.

El software ha sufrido una crisis que data aproximadamente de un cuarto de siglo. En este periodo comenzó a presentarse un fenómeno llamado La Crisis del Software, identificada aproximadamente en los años 60 y fue resultado de la introducción de la tercera generación del hardware.

2.1.1 Causas de la crisis del software.

Los sistemas de información son como cualquier sistema de una empresa en cuanto a que interactúa con otros componentes de la compañía. La tarea de los sistemas de información está en procesar la entrada y producir información.

Los sistemas de información se integran por subsistemas que incluyen el hardware, software y almacenamiento de datos para los archivos y bases de datos. Para que el sistema de información entre en crisis, cualquiera de sus partes debe estarlo.

Durante los primeros años de desarrollo de las computadoras, el hardware sufrió continuos cambios, mientras que el software se contemplaba simplemente como un añadido.

Esta crisis surgió debido a que para los profesionales informáticos el desarrollo del software era un problema muy complicado que con el paso del tiempo empeoraría más y más.

La crisis del software se refiere a un conjunto de problemas que han sido encontrados en el desarrollo del software de computadoras. Estos problemas no están limitados sólo al software que no funciona adecuadamente.

El desarrollo del software se realizaba virtualmente sin ninguna planificación (Pressman, 1993), hasta que las demandas y requerimientos comenzaron a desbordarse y los costos a crecer.

Hacia el final de los años 60, estaba siendo muy claro para los profesionales informáticos el problema que atraía el software.

Las primeras conferencias internacionales sobre ingeniería del software, organizadas por la NATO en 1968 y 1969, pusieron de manifiesto que los grandes proyectos sufrían graves problemas análogos de retrasos en los plazos sobre costes y una gran cantidad de fallas y defectos.

[8] J. Fraude, Erick. Ingeniería del Sotware. Una perspectiva Orientada a Objetos. Ed. Alfa Omega, México 2003. p 104

La crisis del software abarca los problemas que se asocian con la forma de desarrollar el software, cómo mantener un volumen creciente de software existente, cómo satisfacer la demanda creciente de software y cómo satisfacer los requisitos que cada vez son más exigentes por parte del cliente.

Los problemas asociados con la crisis del software se han producido por el propio software y por los errores de los ingenieros responsables del desarrollo del mismo. El éxito del software se mide por la calidad de una única entidad en vez de por varias entidades fabricadas. Si llega a haber alguna falla, existe una alta probabilidad de que se llegue a introducir sin notarlo durante el desarrollo por no haberse detectado durante la prueba.

2.1.2 Acontecimientos sobresalientes de la Crisis del Software.

Durante estos años se ha teorizado, polemizado, propuesto y refutado largamente. He aquí algunos de los acontecimientos más significativos:

- Las conferencias de la NATO en 1968-69, que tuvieron lugar respectivamente en Garmisch y Roma, fueron de utilidad para fijar los problemas de dimensión, complejidad y rendimiento del software. Entre los principales actores hay que mencionar a Ross (M.I.T.), David (Bell Labs), Needham (Cambridge University) y Aron (IBM).
- En 1971 se publicó un artículo Nicklaus With en la revista Communications of the ACM sobre el concepto de modularidad del software. A lo largo de varios años, padres fundadores de la ingeniería del software como Parnas, Myers, Liskov o Zilles discutieron y elaboraron propuestas para el tratamiento de esta cuestión.
- El debate sobre la abstracción de las estructuras de datos y de los procedimientos en los lenguajes de programación lo inició Guttag en 1977 y adquirió magnitud a través de Simula y Smalltalk con los noruegos Nygaard y Dahl, dando origen a lo que, años más tarde, se denominó "orientación a objetos".
- La pugna desarrollada con motivo de los métodos y de los lenguajes estructurados, iniciada en los años 70 y que se prolongó durante más de una década. Aquí se deben recordar los nombres de Jackson, Orr, Warnier y Constantine y los de sus críticos Abrahams, Berry y Clemens, entre otros.
- Se publica un libro en la comunidad del software, "The Mythical Man-Month", escrito por Brooks, el manager del proyecto del OS/360 de IBM, sobre cuestiones prácticas como la dinámica de los equipos de programadores, la escalabilidad del software, los principios de diseño y las técnicas de estimación de proyectos.
- El impacto de los trabajos de la "escuela matemática", en la que se sitúan Mills, Hoare y Dijkstra, partidarios de conducir los procesos intuitivos de la programación hacia modelos rigurosos aplicables a la construcción de soft, basados en la lógica formal y en las matemáticas.
- Los estructuralistas, que formularon propiedades de la complejidad, fiabilidad y mantenibilidad del software a partir del estudio de las

propiedades estructurales de los programas. Entre ellos se puede destacar a Halstead, Mc Cabe, Kernighan, Myers y Plauger.
- Los representantes, intuitivos y formales, cuya influencia metodológica en la modelización de datos es bien conocida, y los impulsores de la denominada cuarta generación (4GL) como Martin, Read y Harmon.
- Las palabras con que se clausuraba la mesa redonda de expertos en la Conferencia Internacional de Ingeniería del Software de 1978: "los problemas de los '80s se parecen mucho a los problemas de los '70s y son deprimentemente similares a los de los '60s". En la conferencia de 1985, Geoffrey Pattie, ministro de estado británico de Industria y Tecnología de la Información, parecía confirmarlo años más tarde: "Para decirlo de forma clara, demasiado software se entrega en condiciones insatisfactorias, muy a menudo tarde, a un coste elevado, con fallos inaceptables".
- En la década de los '90s, la revista Scientific American, intentando explicar la crisis crónica del software concluía: "la persistencia de la crisis es decepcionante". La necesidad histórica de una síntesis de los fundamentos teóricos y de los métodos prácticos de esta joven rama de la ingeniería, ha de conjugarse con el pragmatismo que imponen a la industria las condiciones de nuestra época.

2.1.3 Características de la Crisis del Software.

La crisis del software se caracteriza por varios problemas, los responsables del desarrollo del software se enfocan sobre los aspectos de fondo que son:

- La planificación y estimación de costes frecuentemente muy imprescindible.
- La productividad de la gente del software no corresponde a la demanda de sus servicios.
- La calidad del software es inadecuada.

Como consecuencia, esto ha llevado a sobrepasar los costos en un orden de magnitud. Se ha errado en la planificación en meses o incluso hasta en años. Se hace muy poco para lograr una mejora de productividad de los desarrolladores de software. Los errores en nuevos programas producen en los clientes una gran insatisfacción y falta de confianza.

Tales problemas son las manifestaciones más visibles de otras dificultades del software.

- Hay falta de tiempo para recoger datos sobre procesos de desarrollo del software. Si no hay una indicación sólida de productividad, no se puede evaluar con precisión la eficacia de nuevas herramientas, nuevas técnicas o nuevos estándares.

- La insatisfacción del cliente con el producto terminado es muy frecuente.

- La comunicación entre el desarrollador y el cliente es escasa. Los proyectos de desarrollo del software dan una insuficiente indicación de los requerimientos del cliente.

- La calidad del software es cuestionable.

- El software existente es de difícil mantenimiento, esto se lleva la mayor parte del dinero invertido en el software.

2.1.4 La Ingeniería del Software como solución a la crisis del software

La Ingeniería del Software es un tipo de ingeniería, por lo tanto, tiene el mismo conjunto de responsabilidades sociales que las demás ingenierías. Es el proceso de construir aplicaciones de tamaño o alcance prácticos, en las que predomina el esfuerzo del software y satisfacen los requerimientos de funcionalidad y desempeño.

La Ingeniería de Software surge como disciplina a fines de la década de los sesenta, cuando se presentan los graves problemas existentes en la producción y con el mantenimiento del software, situación que se conoce como la "crisis del software".

Las prácticas de programación y la falta de documentación que se utilizaban en ese tiempo provocaron crisis y determinaron que se pensara en el software como un problema que abordarse de manera sistemática. El modelo básico de esta sistematización corresponde a una variación de un modelo desarrollado en la década de los treinta en los Laboratorios Bell, método que en el ámbito del software se conoce como el ciclo de vida tradicional.

La idea de este ciclo de vida tradicional (Fig. 2.1) es remitir las actividades de desarrollo de software a etapas de desarrollo según el modelo básico de la ingeniería:

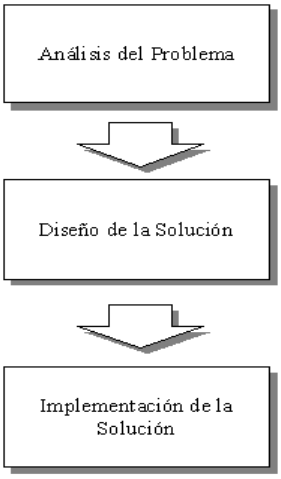


Fig. 2.1 Ciclo de vida tradicional

La primera etapa se trata de hacer una caracterización del problema y del ambiente en que ese problema está inserto. La idea es tener, desde distintos enfoques, una clara visión del problema y de los elementos con que se cuenta para solucionarlo.

El Diseño de la Solución, se orienta a proponer y evaluar distintas posibles soluciones del problema descrito, que su análisis considere, para llegar, finalmente, a la que será la solución a implementar en la etapa siguiente.

La etapa de implementación es aquella que permite concretar, en el ámbito del problema, la solución diseñada en el paso anterior.

Este modelo es acogido en los primeros años de la Ingeniería de Software y adaptado a las condiciones particulares que presenta el software como producto, dio origen a lo que sería una de las primeras metodologías de desarrollo de software, en el sentido que entregaba los lineamientos generales que organizan las actividades a realizar en el proceso de producción.

La solución de la crisis del software es la Ingeniería del Software. La clave está en dar un enfoque de ingeniería al desarrollo del software, junto con la mejora continua de técnicas y herramientas, de tal manera que se trate de combinar métodos completos para todas las fases del desarrollo de un proyecto del software, herramientas CASE que puedan automatizarlas, bloques de construcción más potentes para su implementación, mejoras técnicas para garantizar la calidad del proyecto, y una filosofía predominante para la coordinación, control y gestión de todo el proceso.

El objetivo de la Ingeniería del Software es:

- Mejorar la calidad de los productos de software.
- Aumentar la productividad y trabajo de los ingenieros del software.
- Facilitar el control de procesos de desarrollo del software.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.
- Definir una disciplina que garantice la producción y el mantenimiento de los productos software desarrollados en el plazo fijado y dentro del costo estimado.

2.2 Calidad del Software.

La calidad es un conjunto de propiedades y características de un producto o servicio, que le confieren aptitud para satisfacer necesidades explícitas e implícitas (ISO 8402).

La calidad del software es definida por Pressman (1998) como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente.

2.2.1 Funciones y diseño de la Calidad del Software.

Aunque están desarrolladas por organizaciones excepcionales, las aplicaciones grandes contienen defectos. El software no se desgasta del modo que lo hacen las aplicaciones físicas y se requieren definiciones específicas para la calidad del software.

Una función de calidad, por otro lado consiste en un código que: [9]

- Satisface los requerimientos establecidos con claridad.
- Verifica sus entradas; reacciona de manera predecible a entradas ilegales.
- Se ha inspeccionado de manera íntegra por ingenieros que no son el autor.
- Se ha probado de modo exhaustivo en varias formas independientes.
- Está bien documentado.
- Tiene una tasa de defectos confiable conocida.

Un diseño de calidad casi siempre:

- Se extiende (es fácil manejarlo para proporcionar funcionalidad adicional).
- Evoluciona (se puede adaptar con facilidad a requerimientos diferentes).
- Se mueve (se aplica a varios entornos).
- Es general (se aplica a varias situaciones diferentes).

La meta es especificar los estándares de aceptación y crear productos que satisfagan estas especificaciones. Para hacerlo, se debe saber cómo cuantificar la calidad, cómo especificar metas en términos de estas cantidades, y cómo controlar el avance hacia las metas.

2.2.2 Métricas

La cuantificación es una parte esencial de la ingeniería. Las líneas de código, número de clases, número de defectos fijos por mes y número de funciones por clase son un ejemplo de métricas que se utilizan en la Ingeniería del Software.

Las métricas no pueden separarse de su contexto. Por ejemplo, cuando dos programadores producen diferentes cantidades de código para lograr el mismo grado de confiabilidad, funcionalidad, legibilidad y eficiencia, la versión con el número más pequeño de líneas de código es quizá superior. En otros contextos, más líneas de código pueden indicar una mayor productividad. Sin embargo, tomadas en números grandes de puntos de muestra y usadas junto con otras métricas, las líneas de código pueden ser significativas. Suponiendo que las métricas de confiabilidad están en el nivel requerido y que aumentar las líneas de código refleja una mayor capacidad de la aplicación, cuantas más líneas de código se produzcan por hora, mejor. Debido a la variación de la cobertura entre las métricas, suelen recolectarse varios tipos diferentes.

[9] J. Fraude, Erick. Ingeniería del Sotware. Una perspectiva Orientada a Objetos. Ed. Alfa Omega, México 2003. p 38

Las métricas que casi siempre se incluirán son:

- Cantidad de trabajo realizado, medido físicamente (como líneas de código).
- Tiempo que toma realizar el trabajo.
- Tasa de defectos (defectos por 1000 líneas de código, defectos por página de documentos, etc.).

Con frecuencia deben incluirse clasificaciones subjetivas acerca de calidad del trabajo en una escala de 0 a10.

Los valores previstos o deseados para las métricas se pronostican antes de realizar el esfuerzo, y después se comparan los resultados con los valores pronosticados.

2.2.3 Procesos de aseguramiento de calidad.

Además de la responsabilidad de cada ingeniero de desarrollo, y de la revisión que proporcionen sus colegas, muchas organizaciones identifican un proceso separado de revisión sistemático y exhaustivo, que se conoce como *aseguramiento de calidad* (QA, quality assurance). La función QA incluye revisiones, inspecciones y pruebas.

Como se ilustra en la figura 2.2, el insumo de QA debe buscarse desde el principio de cada proyecto. Es ideal que el QA esté involucrado con garantizar que se usa un proceso sólido y que la documentación se mantiene actualizada. Un representante de QA puede participar con frecuencia en la inspección. De manera ideal, una persona ajena a la organización debe realizar el QA. Muchas compañías son muy pequeñas para este nivel de especialización, en cuyo caso, los ingenieros realizan las funciones de QA para el trabajo de los otros ingenieros.

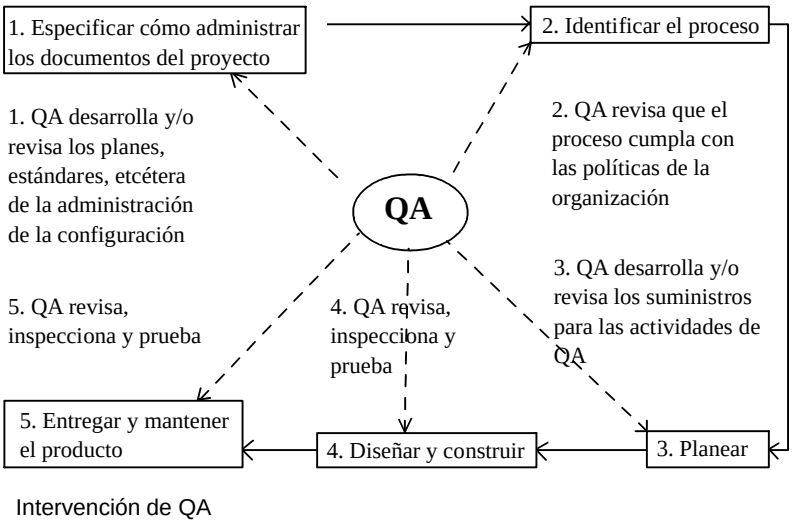


Fig. 2.2 Proceso de QA

"Las desiciones importantes como la selección de la arquitectura no se toman sin desarrollar y comparar alternativas. Las arquitecturas propuestas se examinan de manera exhaustiva en la busca de defectos ya que encontrar un defecto en las primeras etapas del desarrollo significa una recompensa enorme comparado con permitir que uno persista a lo largo del proceso y después tratar de repararlo." [10]

2.2.4 Técnicas de la caja negra y la caja blanca.

Las técnicas de caja negra de QA manejan aplicaciones, o partes de ellas, que ya están construidas. Estas técnicas verifican si el software cumple o no con sus requerimientos. Las técnicas de caja blanca (o caja de vidrio) de QA se aplican a los componentes que forman la unidad que se está probando.

Aunque muchas veces se piensa en las cajas negra y blanca en el contexto de las pruebas, estos conceptos se aplican a varias actividades de aseguramiento de la calidad. Las técnicas de caja blanca requieren que el ingeniero piense en la estructura, forma y propósito del artefacto que examina. Esto incluye usar métodos formales de inspección.

- Inspección

Una inspección es una técnica de caja blanca para asegurar la calidad. Consiste en examinar las partes de proyecto (requerimientos, diseños, código, etcétera) para encontrar defectos. Las inspecciones deben usarse en cuanto se produce la primera documentación del proyecto y se introdujeron para mejorar el código, suelen llamarse inspecciones de código, aunque se haya demostrado que su valor más alto se logra al usarlas temprano en el proceso, mucho antes de producir el primer código.

El concepto de inspección fue establecido por Fagin, quien observó que el autor de un trabajo casi siempre es capaz de reparar un defecto en cuanto sabe que está presente. Entonces, debe aplicarse un proceso que señale al autor los defectos en el trabajo antes de que haga su entrega a la administración. Esto implica que las inspecciones deben ser un proceso entre colegas.

El principio de inspección se puede extender a cuatro reglas:

1. Sólo detección de defectos. Las inspecciones excluyen de modo específico la reparación de defectos. El proceso de reparación se deja al autor y no debe dedicarse tiempo de inspección ni siquiera a sugerencias. Estas deben hacerse fuera de este tiempo.
2. Proceso de colegas. Un grupo de ingenieros de software debe realizar las inspecciones. El trabajo que se somete a inspección debe ser el resultado del mejor esfuerzo del autor, no un borrador o un diseño preliminar. Es una pérdida de recursos que un grupo busque, encuentre

[10] J. Fraude, Erick. Ingeniería del Sotware. Una perspectiva Orientada a Objetos. Ed. Alfa Omega, México 2003. p 283

y describa defectos que el autor hubiera encontrado con un esfuerzo razonable.

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

    - Moderador. Es responsable de ver que se lleve a cabo la inspección de modo apropiado.
    - Autor. Es responsable del trabajo en sí, y repara los defectos encontrados.
    - Lector. Es responsable de 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 lo decida el equipo.

4. Preparación completa. Se requiere que los participantes en inspecciones se preparen al mismo nivel de detalle que el autor. Las inspecciones no son revisiones, vistazos de la gerencia, ni sesiones educativas. Los inspectores deben trabajar al mismo nivel de detalle que el autor.

Los pasos siguientes se requieren para ejecutar una inspección:

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

Si es necesario, debe organizarse una junta de visión general para explicar la unidad sometida a la inspección.

La siguiente etapa consiste en la preparación y es donde los inspectores revisan el trabajo con todo detalle en sus propios escritorios e introducen los defectos encontrados en una base de datos (accesible desde la red) junto con las descripciones y clasificaciones. Esto ayuda a prevenir la duplicidad y minimiza el tiempo innecesario de juntas.

Una vez que todos los participantes están preparados, se lleva a cabo la junta de inspección, en la cual, los participantes toman sus roles asignados.

Por lo común, el autor es capaz de reparar todos los defectos. Esta es la etapa en que se debe tal vez volver a hacer el trabajo, sin embargo, si la junta de inspección decide que los defectos son tan fuertes que requieren una nueva inspección, entonces la entidad se recicla a través del proceso.

Si los defectos se deben a malos entendidos o errores conceptuales, quizá sea necesario convocar a una junta separada de análisis causal en la que se discutan estas causas.

La junta de seguimiento final es breve; en ella el moderador y el autor confirman que los defectos se repararon. No se supone que esta sea una revisión detallada del moderador. La responsabilidad de la reparación es del autor, que es el encargado del trabajo.

2.2.5 Estándar de IEEE para planes de aseguramiento de la calidad del software

Las listas de verificación de cosas que hacer y buscar son en particular útiles para asegurar que están cubiertas todas las bases de la calidad; de ahí el uso de los estándares existentes.

El estándar IEEE 730-1989 se refiere al plan de aseguramiento de la calidad del software. Gran parte de este estándar está dirigido a proyectos grandes, pero también puede usarse en los pequeños y el estándar ayuda a recordar todos los factores que deben incluirse.
El documento especifica:

- Quién será responsable de la calidad; una persona, un gerente, un grupo, una organización, etc.

- Qué documentación se requiere.

- Qué técnica se usará para asegurar la calidad; inspecciones, demostración de qué es correcto, pruebas, etc.

- Qué procedimientos se seguirán para administrar el proyecto; reuniones, auditorias, revisiones, etc.

2.3 Componentes de la Ingeniería del Software.

La reutilización es una característica muy importante para lograr un software de buena calidad. Los componentes de software son construidos por medio de lenguajes de programación con un vocabulario limitado, una gramática definida y reglas bien formadas de sintaxis y semántica.

En las décadas de 1980 y 1990, dos tendencias fueron las que dominaron la ingeniería del software, una de ellas fue el crecimiento explosivo de aplicaciones, incluyendo las asociadas con Internet. La otra es el florecimiento de nuevas herramientas y paradigmas.

Sin embargo, a pesar de la llegada de nuevas tendencias, las actividades básicas requeridas para la construcción del software han permanecido estables.

2.3.1 Procesos de la Ingeniería del Software.

El proceso de la Ingeniería del Software se puede definir como un conjunto de etapas, actividades, métodos y prácticas que guían a los ingenieros en la producción de Software con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad.

El proceso Software es llevado a cabo por personas que usan: herramientas, técnicas y metodologías.

Los factores que determinan el proceso software son: Política, Tecnología, Economía e Ingeniería del Software, costumbre, cultura, estándares y obligaciones contractuales.

Entre los beneficios de un buen proceso tenemos:

- Mejora el uso de los recursos humanos.
- Proporciona un mecanismo para aprender experiencias de otros.
- Reduce repetición de trabajos.
- Proporciona más tiempo para gastar en los problemas que requieren energía de creatividad.
- Incrementa la probabilidad de introducir tecnología con éxito.
- Proporciona mejora continuada de capacidades de producción del software.

En el proceso de la Ingeniería del Software se debe tener en cuenta una serie de componentes a saber:

- Formalismos o teorías base para el desarrollo de los demás componentes.
- Componentes de la administración del proyecto.
- Técnicas y metodologías de desarrollo y mantenimiento.
- Herramientas CASE.

Todas permiten cubrir el ciclo de vida del software: requerimientos, especificaciones, diseño, realización y mantenimiento.

Además de estos componentes, otros dos fundamentales, son el usuario conocedor del problema y de las necesidades y el ingeniero de software conocedor y dominador de los componentes técnicos y capaz de plasmar en un producto software funcionando en una organización, la solución a las necesidades planteadas por el usuario.

Se puede considerar a la Ingeniería del Software como una colección de tecnologías entrelazadas que deberán ser consideradas como un sistema.

Un objetivo de décadas ha sido el encontrar procesos o metodologías predecibles y repetibles que mejoren la productividad y la calidad.

a) Pasos del proceso

La ingeniería de software requiere llevar a cabo muchas tareas, sobre todo las siguientes:

Análisis de requisitos: la primera etapa para crear un producto de software es extraer sus requerimientos. Se requiere tener la experiencia para reconocer cuando los requisitos son incompletos.

Especificación: es describir con detalle al software de forma escrita. Las especificaciones son muy importantes para las interfaces externas, que deben permanecer estables.

Diseño y arquitectura: se refiere a determinar la manera en que funcionará en forma general sin entrar en detalles. Se debe tomar en cuenta consideraciones de la implementación tecnológica, como el hardware, la red, etcétera.

Programación: es transcribir un diseño a código, esto puede ser la parte más obvia del trabajo de ingeniería de software, pero no es necesariamente la porción más larga.

Prueba: consiste en hacer varias pruebas para verificar que el software realice correctamente las tareas indicadas en la especificación. Se puede revisar por separado cada módulo del software, y luego probarlo de forma integral.

Documentación: es la realización del manual de usuario, y tal vez de un manual técnico con el fin del mantenimiento futuro y nuevas aplicaciones al sistema.

Mantenimiento: se refiere a mantener y mejorar el software para enfrentar errores o problemas que con el tiempo sean descubiertos y la integración de nuevos requerimientos. Esto puede llevar más tiempo incluso que el desarrollo inicial del software. Alrededor de 2/3 de toda la ingeniería de software tiene que ver con dar mantenimiento La mayor parte consiste en extender el sistema para hacer nuevas aplicaciones.

2.3.2 Modelos del proceso del Software.

Todo el desarrollo del software se puede caracterizar como un ciclo de resolución de problemas que se divide en cuatro etapas distintas:

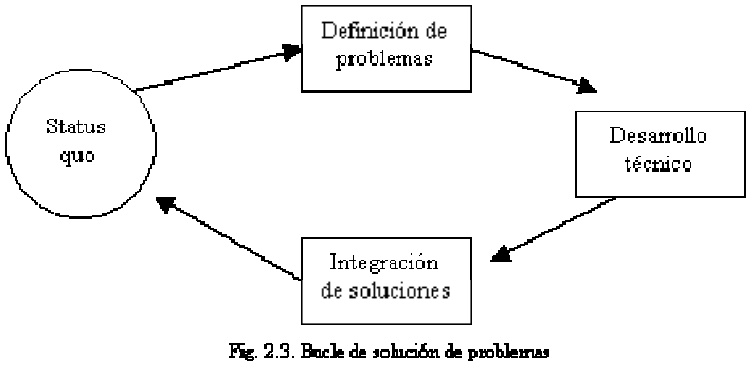


Fig. 2.3. Bucle de solución de problemas

Fig. 2.3 Bucle de solución de problemas

El status quo repre

Modelo lineal secuencial: Es denominado también ciclo de vida básico o modelo de cascada.

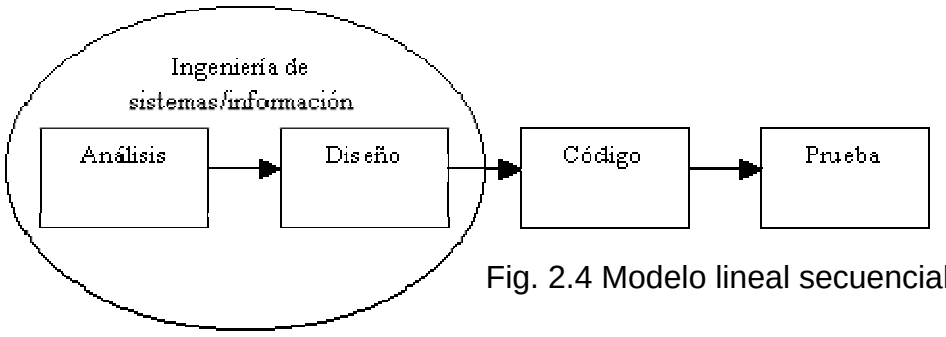


Fig. 2.4 Modelo lineal secuencial

Modelo de construcción de prototipos: Ofrece un enfoque a través del paradigma de construcción de prototipos.

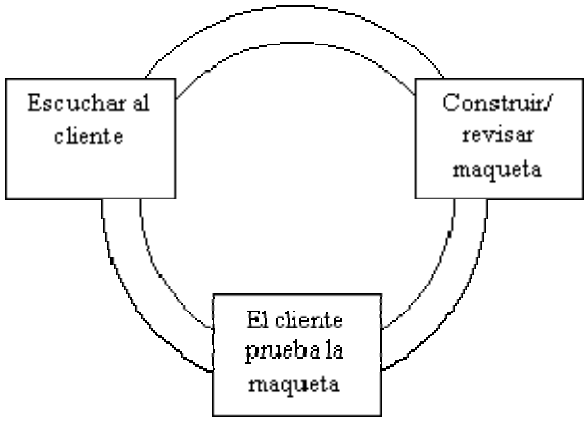


Fig. 2.5 Modelo de construcción de prototipos

Modelo DRA: El Desarrollo Rápido de Aplicaciones (DRA) (Rapid Application Development) es un modelo de proceso de desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. Es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido, utiliza un enfoque de construcción que se basa en componentes.

Modelos de proceso evolutivo del software: Los requisitos de gestión y de productos a menudo evolucionan conforme al desarrollo, haciendo que el camino que lleva al producto final sea irreal; las estrictas fechas del mercado imposibilitan el término de un producto completo, por esto, se debe introducir una versión limitada para cumplir la presión competitiva y de gestión; se comprende perfectamente el conjunto de requisitos de productos centrales o del sistema, pero todavía hará falta definir los detalles de extensiones del producto o sistema.

- Modelo incremental: Combina elementos del modelo lineal secuencial con la filosofía interactiva de construcción de prototipos.

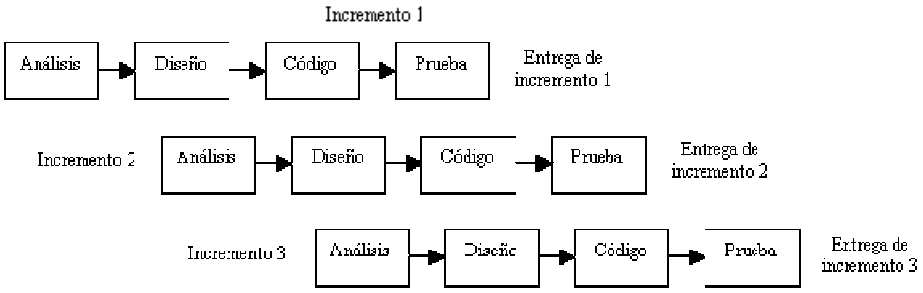


Fig. 2.6 Modelo Incremental (a)

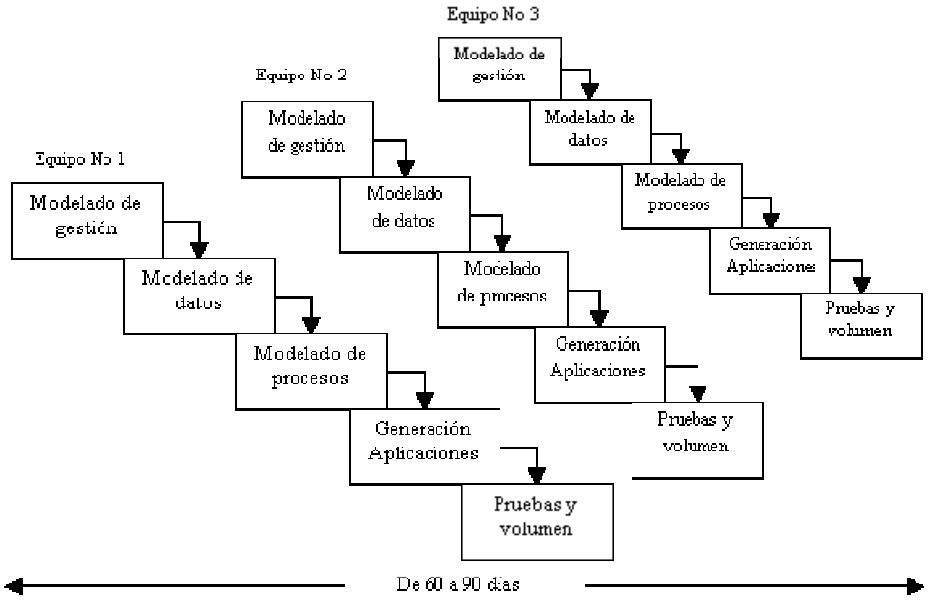


Fig. 2.7 Modelo Incremental (b)

- Modelo en espiral: En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas de la ingeniería del sistema.

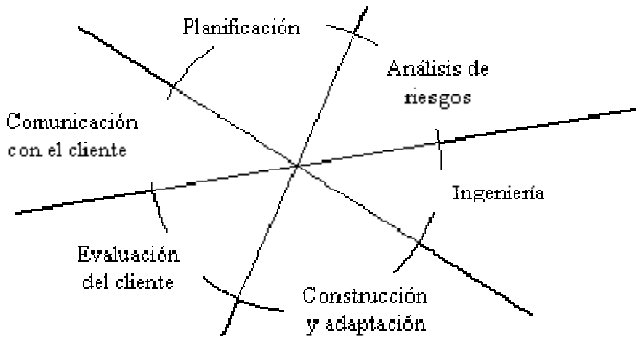


Fig. 2.8 Modelo en espiral.

2.3.3 Herramientas del proceso de desarrollo.

Es un conjunto de herramientas software para automatizar las tareas del desarrollo de software que se utiliza en una o cualquiera de las fases de desarrollo de sistemas de información, incluyendo análisis, diseño y programación. Por ejemplo las herramientas de diagramación ayudan en las fases de análisis y diseño mientras que los generadores de aplicaciones aceleran la fase de programación.

2.3.3.1 Tecnología CASE y entornos de desarrollo.

La tecnología CASE corresponde a la Ingeniería del Software asistido por computadora.

Las herramientas CASE proporcionan métodos automáticos para diseñar y documentar las técnicas tradicionales de Programación Orientada a Objetos. La meta es 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.

Entre los objetivos de la tecnología CASE tenemos:

- Aumentar la productividad en el desarrollo.
- Dar calidad a los productos desarrollados.
- Reducir el coste del software.
- Automatizar los chequeos de errores.
- Acelerar 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.
- Ingeniería hacia atrás.

Entre los tipos de herramientas CASE tenemos:

- Herramientas Front-end.
    - Herramientas de planificación y definición de requisitos.
    - Herramientas de análisis y diseño.
    - 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.

Conjunto de herramientas integradas que interactúan unas con otras de forma consistente.

- Conforme a un conjunto de estándares.
- Ante el usuario conforman un bloque único sin funciones o mensajes duplicados.

En lo que respecta 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 gráfica para dibujar diagramas estructurados.
- Una enciclopedia para almacenar y administrar toda la información del sistema software.
- 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.

Un elemento clave de ayuda en las herramientas CASE es la enciclopedia, el cual es un mecanismo para almacenar y organizar toda la información relativa a un sistema software. Incluye información relativa a Planificación estratégica, Análisis, Diseño, Implementación, Administración del proyecto.

La enciclopedia es un punto clave para una alta productividad, permite obtener información para los realizadores cuando se necesita y directamente utilizable. Es el único lugar donde se alberga toda la información del sistema, de forma consistente y disponible para quien la necesite.

Es más que un diccionario, almacena tipos de información del sistema, interrelación entre componentes de información, reglas para usar o procesar los componentes.

Las funciones que realiza son: almacenamiento, acceso, actualización, análisis e informes sobre la información del sistema.

Además, administra la información del sistema (técnica y administrativa), mantiene toda la información necesaria para crear, modificar y mantener un sistema software, e incluye información relativa a:

- Problema a resolver.
- Dominio del problema.
- Sistema solución emergente.
- Proceso software a seguirse (método).
- Historia y recursos del proyecto.
- Contexto organizativo.

Las herramientas tienen escaso valor si no soportan una metodología y una escasa repercusión productiva si antes o simultáneamente no se implanta una metodología.

Las estaciones de trabajo CASE constituyen un entorno completo que incluye hardware y software; además, proceso de textos, almacenamiento y obtención de la información, correo electrónico y funciones calendario; proporcionan soporte máximo para el realizador software individual y el objetivo primario es incrementar la productividad, y en segundo lugar mejorar la calidad.

2.4 UML (Lenguaje Unificado de Modelado).

UML es un lenguaje de modelado y no un método. La mayoría de los métodos consisten en un lenguaje y en un proceso para modelar. 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.

2.4.1 Orígenes y beneficios del modelado con UML.

Este Lenguaje Unificado de Modelado o UML es el sucesor de la oleada de métodos de análisis y diseño orientados a objetos (OOA&D) que surgió a finales de la década de 1980 y principios de la siguiente. El UML unifica, sobre todo, los métodos de Grady Booch, Jim Rumbaugh (OMT) e Ivar Jacobson (“Los tres amigos”), pero su alcance llegará a ser más amplio. En estos momentos el UML está en pleno 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. Muchos se preguntaban cómo se adecuarían los métodos de diseño a un mundo orientado a objetos. Los métodos de diseño se habían vuelto muy populares en el desarrollo industrial durante las décadas de 1970 y 1980. Muchos pensaban que las técnicas para ayudar al buen análisis y diseño eran también importantes en el desarrollo orientado a objetos.

Los libros clave sobre el análisis orientado a objetos y los métodos de diseño aparecieron entre 1988 y 1992:

- Rally Shlaer y Steve Mellor escribieron un par de libros (1989 y 1991) 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 también 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 había trabajado mucho con Racional Software, desarrollando sistemas en Ada. En sus libros se daban varios ejemplos.

- Jim Rumbaugh dirigió un equipo en los laboratorios de investigación de General Electric, cuyo resultado fue un popular 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).

Cada uno de los autores antes mencionados dirigía informalmente un grupo de profesionales que estaban de acuerdo con sus ideas. Todos estos modelos eran muy similares; sin embargo, tenían entre sí gran cantidad de diferencias menores. Los mismos conceptos básicos aparecían con denominaciones diferentes.

Grady y Jim proclamaron: “la guerra de los métodos ha terminado; la ganamos nosotros”, declarando, en esencia, que iban a lograr la estandarización a la manera de Microsoft. Y 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 otro 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.

“Hoy en día es necesario contar con un plan bien analizado. Un cliente tiene que comprender que es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades.” [11]

UML también permite organizar el proceso de diseño más fácilmente, de manera que pueda ser comprendido por analistas, clientes, desarrolladores y los demás miembros del equipo que está relacionado con el proyecto.

UML está conformado por elementos gráficos combinados para conformar diagramas y cuenta con diversas reglas para combinar dichos elementos, ya que como mencioné anteriormente UML es un lenguaje.

En los diagramas de UML se describe gráficamente lo que hará el sistema, pero no describe cómo se implementará.

Para el desarrollo de software orientado a objetos no basta usar un lenguaje con las mismas características. También se necesitará realizar un análisis y diseño orientado a objetos.

Según los mismos diseñadores del lenguaje UML, este tiene como fin modelar cualquier tipo de sistemas (no solamente de software), usando los conceptos de la orientación a objetos. Además, este lenguaje debe ser entendible para los humanos y máquinas.

El UML consta de todos los elementos y diagramas que permiten modelar los sistemas en base al paradigma orientado a objetos. Los modelos orientados a objetos cuando se construyen en forma correcta, son fáciles de comunicar, cambiar, expandir, validar y verificar. Este modelado en UML es flexible al cambio y permite crear componentes plenamente reutilizables.

Algunos diagramas que conforman UML son los siguientes:

2.4.2 Diagrama de Casos de Uso.

Un diagrama de casos de uso describe la secuencia de eventos de un actor que utiliza un sistema para completar un proceso, es una técnica de aciertos y errores para obtener los requerimientos desde el punto de vista del usuario. Su función principal es la captura de los requerimientos del sistema, describir la funcionalidad y los actores que intervienen en él.

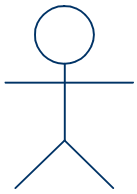
- Representa gráficamente la funcionalidad del sistema, cómo sería vista por el usuario.

- Cómo el cliente trabajará con el sistema.

[11] Schmuller, Joseph. Aprendiendo UML en 24 horas. Ed. Pearson Educación, México 2000. p 6

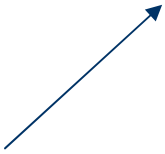
- Están expresados desde el **punto de vista del actor**.

El diagrama de casos de uso consta de:



Actor

Casos de uso



Relaciones (usa, extiende)

Actor: Agente externo que utiliza el sistema para realizar una tarea no necesariamente representa a una persona en particular, sino más bien, representa quién realiza las operaciones identificadas.

Casos de Uso: Es una operación o tarea específica que se realiza tras una orden de algún agente externo, sea un actor o desde otro caso de uso.

Relación extend: Se usa la relación cuando se tiene un caso que es similar a otro pero que es más especializado.

Relación usa: Ocurre cuando se tiene una porción de comportamiento similar en más de un caso de uso y no se quiere copiar la descripción de tal conducta. Es frecuente que no haya un actor que esté asociado con el caso de uso común. Si lo hay, la relación usa no considera que esté llevando a cabo los demás casos de uso.

La realización de un diagrama de casos de uso está basada en los siguientes elementos:

- Enunciado general del problema.
- Entrevista no dirigida.
- Entrevistas dirigidas.
- Análisis de la tarea

Puede existir jerarquización entre actores, donde un actor puede ejecutar todos los casos que lleve a cabo otro actor.

El propósito del diagrama de casos de uso es especificar el contexto y la captura los requerimientos de un sistema, manejar la implementación y generar casos de prueba.

2.4.3 Diagramas de Secuencia.

En un sistema funcional los objetos interactúan entre sí, este diagrama muestra la mecánica de interacción con base en tiempos.

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

Cada mensaje se 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, y se puede mostrar la autodelegación, que es un mensaje que un objeto se envía a sí mismo, regresando la flecha de mensaje de vuelta a la misma línea de vida.

2.4.4 Diagramas de Colaboración.

En los diagramas de colaboración, los objetos ejemplo se muestran como iconos. Las flechas indican, como en los diagramas de secuencia, los mensajes enviados dentro del caso de uso dado. La secuencia se indica numerando 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 entre ellos los objetos y emplear la disposición para sobreponer paquetes u otra información.

2.4.5 Diagrama de Clases.

Estos diagramas colaboran en lo referente al análisis, constan de atributos y métodos y facilitan las representaciones a partir de las cuales los desarrolladores podrán trabajar, además de que permiten al analista hablar con los clientes en su terminología propia, lo cual hace posible que los clientes indiquen detalles importantes de los problemas que requieren ser resueltos.

El diagrama de clase, además de ser de uso extendido, también está sujeto a la más amplia gama de conceptos de modelado. Describe los tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos.

Los diagramas de clase también muestran los atributos y operaciones de una clase y las restricciones a que se ven sujetos, según la forma en que se conecten los objetos. Describe gráficamente las características de las clases de software en una aplicación (recordemos que Clase es la unidad básica que encapsula información).

Nomenclatura: El Nombre de Clases Inicia con mayúscula; si el nombre tiene más de una palabra, se unen.

Clase ✓

NombreClase ✓

Artículo ✗

Atributos: Dependiendo de la comunicación y visibilidad entre ellos y el medio que los rodea, están divididos en:

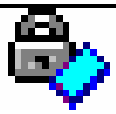
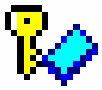
Símbolo	Nombre	Descripción
	**Atributos Públicos (+)**	Accesibles desde todos lados, tanto dentro como fuera de la clase.
	**Atributos Privados (-)**	Sólo es accesible desde dentro de la clase (sólo sus métodos los podrán accesar).
	**Atributos Protegidos (#)**	No se puede accesar desde fuera de la clase, pero si por métodos de la clase y subclases.

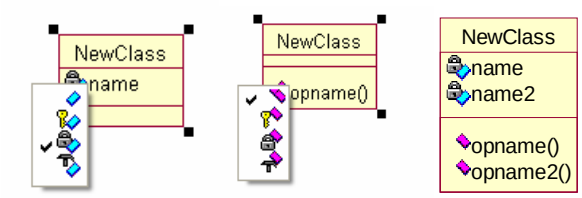
Tabla 2.1 Atributos

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

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

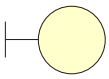
Tabla 2.2 Métodos

Ejemplo de clases de diseño:



Estereotipos de clases de diseño:

Clases "Boundary" (Interfaces)



Boundary

Clases de "Control" (Métodos)

Control

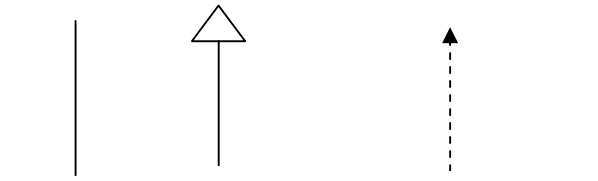
Clases "Entity" (Atributos y Métodos)

Entidad

Modelo Conceptual: Representa los conceptos significativos en el dominio del problema real. Define y modela los aspectos más importantes acerca de la información que la empresa necesita obtener y las relaciones entre dicha información. Describe la estructura del sistema, independiente del software que se use. Se dibuja un diagrama que represente los conceptos del dominio que se está estudiando, estos conceptos se relacionan de manera natural con las clases que los implementan. Se debe dibujar sin importar el software con que se implementarán, por lo cual se pueden considerar como independientes del lenguaje.

Asociación:

- Permite relacionar objetos que colaboran entre sí.
- Es una relación entre dos conceptos que indica alguna conexión entre ellos.



Asociación Herencia Dependencia o instancia

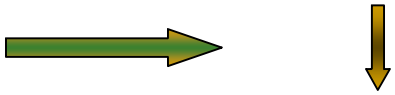
En el modelo conceptual, ayuda a mejorar la comprensión del dominio del problema.

La asociación se representa como una línea entre conceptos. Permite asociar conceptos que colaboran entre sí.

La herencia indica que una subclase hereda los métodos y atributos especificados por una superclase, por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected).

La dependencia o instancia representa una clase que es instanciada (dependiente de otra clase).

Nomenclatura para asociaciones: Los nombres de Asociaciones comienzan con mayúsculas. Una frase se construye con guiones. La dirección por omisión en que debe leerse el nombre de la asociación es:



Multiplicidad: Es parte de la asociación (se asignan en sus extremos) e indica cuántas instancias de la clase A se asocian a la clase B.

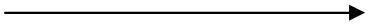
2.4.6 Diagrama de Estados.

También se le conoce como motor de estados y describe los diversos estados en los que puede estar un cierto objeto. Este diagrama junto con el de secuencia representa información estática.

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

Se 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.

Los elementos que lo integran son:



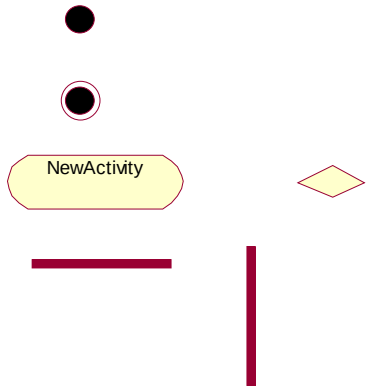
Línea de transición

El diagrama de estados permite a los analistas, diseñadores y desarrolladores comprender el comportamiento de los objetos del sistema. Los desarrolladores, en particular, deben saber la forma en que los objetos se supone que se comportarán, ya que ellos son quienes tendrán que establecer tales comportamientos en el software. Aseguran que no se tenga que adivinar lo que se supone harán los objetos.

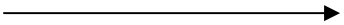
2.4.7 Diagramas de Actividades.

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 mostrando una secuencia de pasos, procesos, puntos de decisión y bifurcaciones.

El diagrama de actividades permite seleccionar el orden en que se harán las cosas. Simplemente dice las reglas esenciales de secuenciación que hay que seguir. Es una parte integral del análisis del sistema.
Los elementos que lo integran son:



Distribuidor horizontal y vertical



Transición

Decisiones Mutuamente excluyentes: La condición va dentro de corchetes junto a la ruta correspondiente.

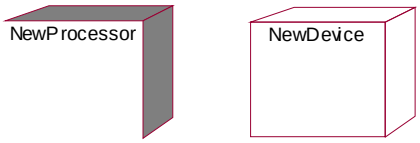
Rutas concurrentes: Se pueden separar actividades que se ejecutan al mismo tiempo(es decir, de forma concurrente) y luego se unan. Para representar esta división se usa una línea gruesa perpendicular y las rutas partirán de ella. Para representar la unión, ambas rutas apuntarán a otra línea gruesa paralela a la anterior.

2.4.8 Diagramas de Distribución.

Este diagrama muestra la arquitectura física de un sistema, en él se puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrará en cada máquina. Las máquinas se representan por un cubo y las interacciones entre las computadoras se representan con líneas que conectan a los cubos.

El diagrama de distribución muestra la topología del hardware

Nodos Conexión



Este diagrama constituye parte de la especificación de la arquitectura y es desarrollado por ingenieros de redes, ingenieros de sistemas.

3 Procesos de desarrollo en la ingeniería de software.

El proceso de desarrollo de software es el proceso en el que se traducen las necesidades del cliente en requerimientos del software y de ahí se transforman en diseño y se implementa en el código, el cual se prueba y se certifica su uso. El proceso de desarrollo define cómo alcanzar el objetivo final.

3.1 Definición de proceso de desarrollo.

El proceso de desarrollo de software requiere un conjunto de conceptos, una metodología y un lenguaje propio; comprende como mínimo cuatro grandes fases: concepción, elaboración, construcción y transición.

- La concepción define el alcance del proyecto y desarrolla un caso de negocio.
- La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura.
- La construcción crea el producto.
- La transición transfiere el producto a los usuarios.

El objetivo de un proceso de desarrollo es mejorar la calidad del software (en todas las fases por las que pasa) mediante una mayor transparencia y control sobre el proceso. Se debe producir lo esperado en el tiempo esperado y con el coste esperado.

Es obligación del proceso de desarrollo supervisar y lograr que esas medidas se lleven a cabo en cada desarrollo para aumentar la calidad.

En los últimos tiempos la cantidad y variedad de los procesos de desarrollo ha aumentado de forma impresionante. Se podría decir que en estos últimos años se han desarrollado dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y los métodos ligeros. La diferencia fundamental entre ambos es que mientras los primeros intentan conseguir el objetivo común por medio de orden y documentación, los segundos (también llamados métodos ágiles) tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso. A continuación se definen algunos de los procesos para el desarrollo del software.

3.1.1 RUP (Racional Unified Proccess).

RUP es un proceso pesado y es uno de los procesos más generales del los existentes actualmente, ya que en realidad está pensado para adaptarse a cualquier proyecto, y no sólo de software.

Como se mencionó anteriormente cuando se realiza un proyecto siguiendo un proceso RUP se divide en cuatro fases:

1. Intercepción (puesta en marcha)
2. Elaboración (definición, análisis, diseño)
3. Construcción (implementación)
4. Transición (fin del proyecto y puesta en producción)

En cada una de las fases se ejecutan una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada iteración seguirá un modelo de cascada o para los flujos de trabajo que requieren las nuevas actividades.

RUP define nueve actividades a realizar en cada fase del proyecto

1. Modelado del negocio
2. Análisis de requisitos
3. Análisis y diseño
4. Implementación
5. Test
6. Distribución
7. Gestión de configuración y cambios
8. Gestión del proyecto
9. Gestión del entorno

Y el flujo de trabajo (*workflow*) entre ellas en base a los llamados diagramas de actividad. El proceso define una serie de roles distribuidos entre los miembros del proyecto y que definen las tareas de cada uno y el resultado que se espera de ellos.

Fig. 3.1 Flujo de trabajo RUP

RUP está basado en casos de uso para la descripción de lo que se espera del software y se orienta a la arquitectura del sistema, documentándose lo mejor posible, basándose en UML (*Unified Modeling Language*) como herramienta principal. En la figura 3.1 se observa el flujo de trabajo del RUP.

RUP es un proceso amplio y muy general, por lo que antes de usarlo habrá que adaptarlo a las características de la empresa.

3.1.2 XP (Extreme Programming).

XP (Programación Extrema) es una metodología que se recomienda para mejorar la velocidad del desarrollo de un software que fundamente su desarrollo en casos de prueba y en la experiencia del usuario.

Es muy importante que se utilice adecuadamente una metodología para administrar y garantizar el cumplimiento de los requerimientos del proyecto especificados desde un inicio.

XP intenta minimizar la complejidad del software mediante un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción. También intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo, que debe estar capacitado para contestar correctamente a las preguntas del mismo.

XP utiliza *UserStories*[12] como base del software a desarrollar. A partir de las UserStories y de la arquitectura perseguida se crea un plan de *releases* (*liberación* o *entrega* del software) entre el equipo de desarrollo y el cliente.

Para cada liberación se discuten los objetivos de la misma con el representante del cliente y se definen las iteraciones necesarias para cumplir con los objetivos de la liberación del software.

El resultado de cada iteración se proporciona al cliente para que lo analice y juzgue. De acuerdo a su opinión se definen las siguientes iteraciones del proyecto, y si el cliente no está contento se adaptará el plan de liberación de software e iteraciones hasta que quede conforme y conceda su aprobación.

La funcionalidad concreta del software sólo se escribe cuando las pruebas para su corrección estén preparadas.

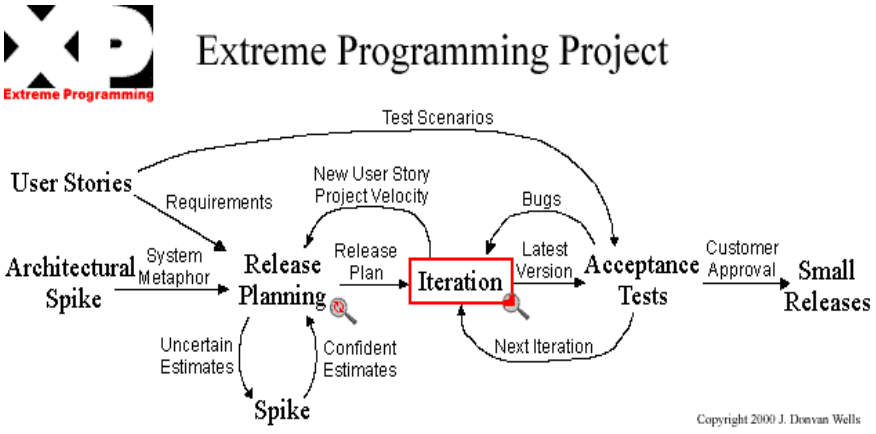


Fig. 3.2 Vista general de XP

[12] UserStories son historias que escribe el cliente y describen escenarios sobre el funcionamiento del software, que pueden describir el modelo, dominio, etc. http://www.agilemodeling.com/essays/agileModelingXP.htm

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

Al contrario que muchos otros métodos, el código pertenece al equipo en completo, no a un programador o pareja, de forma que cada programador tiene derecho y la posibilidad de cambiar cualquier parte del código en cualquier momento si así lo necesita, dejándose en todo caso las mejoras orientadas al rendimiento para el final. Las parejas se rotan cíclicamente a lo largo del proyecto, en cuanto a los componentes de la misma como en las partes del software que desarrollan, así cada componente del equipo aprende como trabaja el resto.

El objetivo ideal es que cada componente del equipo trabaje al menos una vez con cada uno de los demás integrantes y con cada componente de software, y así el equipo completo tiene el conocimiento de toda la aplicación.

En XP, una gran flexibilidad y capacidad de configuración sólo será implementada cuando sea necesaria para cumplir los requerimientos de la liberación del software. 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.

3.1.3 PSP (Personal Software Process).

Humphrey definió con destreza qué aptitudes debe tener un ingeniero de software competente. El resultado es el proceso personal de software (PSP) que proporciona métodos detallados para estimar y planear, muestra a los ingenieros cómo dar seguimiento a su desempeño contra estos planes y explica cómo los procesos definidos pueden guiar su trabajo. Tiene el propósito de desarrollar hábitos de programación, en especial en cuanto a la medición. Supone que el ingeniero ya posee los conocimientos de lenguajes de programación.

El PSP se divide en etapas graduales de crecimiento llamadas PSP0, PSP1, PSP2 y PSP3.

PSP0: Proceso Base PSP0 acepta estudiantes practicantes y sus prácticas de desarrollo actuales, pero requiere que el estudiante mantenga un registro del tiempo dedicado a trabajar en un proyecto, así como de los defectos y tipos encontrados.

El PSP0 aumenta con el PSP0.1, que requiere que el estudiante establezca una manera estándar de definir una “línea de código” y un marco de trabajo dentro del cual el individuo puede observar maneras de mejorar su proceso de desarrollo.

PSP1: Proceso de Planeación Personal El PSP1 está diseñado para ayudar al ingeniero a entender la relación entre el tamaño de los programas y el tiempo que toma desarrollarlos. Su propósito es proporcionar un marco de trabajo ordenado dentro del cual el individuo pueda realizar estimaciones, hacer compromisos, evaluar el estado y registrar los resultados. Una y otra vez, los gerentes preguntan a los ingenieros de software ¿Cuánto tomará hacer esto? El PSP ayuda a los ingenieros a contestar esta pregunta en forma realista.

PSP1 agrega las siguientes aptitudes:

- Aptitud para estimar el tamaño.
- Marco de trabajo para informar los resultados de las pruebas.

El PSP1 aumenta a PSP1.1, el cual agrega la habilidad de realizar tareas de programación del plan y tiempos.

PSP2: Proceso de Administración de la Calidad Personal El PSP2 está diseñado para ayudar a los ingenieros a manejar de manera realista y objetiva los defectos de programación. La idea es estimar tantos defectos como sea posible antes de someter el programa a una inspección formal.

El PSP2 agrega revisión personal del diseño y del código.

El PSP2 aumenta a PSP2.1, que agrega un marco de trabajo y lista de verificación para asegurar que se completen los diseños.

PSP3: Proceso Personal Cíclico. El PSP3 está diseñado para escalar el PSP para manejar las unidades de código grandes (en miles de líneas) dividiendo un programa grande en pequeños incrementos.

PSP3 agrega la aplicación de PSP a cada incremento para producir una alta base de calidad para los incrementos sucesivos y el uso de pruebas de regresión para asegurar que las pruebas diseñadas para los incrementos anteriores todavía son buenas en los nuevos incrementos.

3.1.4 TSP (Team Software Process).

Desde 1990, Watts Humphrey reportó resultados alentadores al establecer las metas de madurez y los procedimientos para el equipo de software. Llamó a este proceso Team Software Process. Los objetivos de TSP son:

- Formar equipos autodirigidos 3 a 20 ingenieros para el desarrollo de proyectos grandes.
- Establecer sus propias metas.
- Establecer sus propios procesos y planes.
- Rastrear el trabajo.
- Mostrar a los gerentes cómo administrar equipos.
- Orientar.
- Motivar.

- Apoyar el desempeño más alto.

El TSP hace hincapié en la iniciativa del equipo y la interacción de abajo arriba que alienta un mayor grado de profesionalismo entre los ingenieros de software. Por ejemplo, Humphrey establece que no es profesional que los ingenieros proporcionen a la administración tiempos que no podrán cumplirse, aun cuando se les pida que lo hagan. En esas situaciones aconseja la negociación. También es digno de hacerse notar el énfasis de TSP en la orientación de la administración externa a los equipos. Se espera que la administración no sólo de órdenes y especifique tiempos de entrega, sino proporcione guía, herramientas y otros recursos necesarios.

4 Análisis del sistema.

Una vez repasados los conceptos teóricos en capítulos anteriores se pasará analizar el problema que se tiene con el apoyo de diagramas utilizados en la metodología UML.

A continuación el enunciado general del problema.

4.1 Enunciado general.

El proyecto será un sistema que permita a una empresa de desarrollo de sistemas llevar un control de las fallas que se puedan presentar en sus productos; así como requerimientos para nuevas funcionalidades solicitados por clientes.

4.2 Problemática actual.

El problema de mayor importancia es que en muchas ocasiones no se lleva una bitácora de todos los requerimientos que solicita un cliente y después del desarrollo de dichos requerimientos el cliente solicita nuevas funcionalidades o no queda satisfecho con el resultado obtenido. Esto provoca que se tenga que invertir más tiempo en la modificación de requerimientos o que se tengan que realizar algunos otros que no estaban planeados.

4.3 Requerimientos funcionales.

Con este sistema se pretende que se mantenga un registro de todos y cada uno de los requerimientos y fallas registradas por cada uno de los clientes. Los puntos que deberá cubrir son los siguientes:

1. Será necesario que posea un módulo para los clientes con los que se cuenta actualmente y que se pueda incluir nuevos clientes o prospectos.

2. Un módulo para levantar los posibles requerimientos de un cliente, llevando registro de la fecha, detalle y prioridad del requerimiento así como el usuario que lo solicita.

3. Un apartado más para que los clientes puedan hacer mención de los posibles fallos que se les presente en alguno de los productos que ha adquirido. Este apartado debe contener también una fecha del reporte de falla, detalle de la misma, prioridad, el usuario que levanta el requerimiento, el producto en el que se presentó la falla y el sistema operativo sobre el que se implementó la aplicación.

4. El sistema también debe tener una sección de respuesta al cliente donde se le indicará la posible solución a su problema en caso de que no requiera asistencia del personal de la empresa. En caso de que el problema tenga que ser resuelto por alguno de los consultores se deberá dar un tiempo aproximado de solución al problema o el tiempo de desarrollo para un nuevo requerimiento.

5. También debe ser posible que el cliente vea las respuestas proporcionadas por los expertos de la empresa en sistemas ya sea para la solución de un problema o información sobre alguna petición realizada.

6. Asimismo es necesario que se le proporcione al cliente un estatus del avance de determinado requerimiento o reporte de falla emitido.

7. Dentro de los parámetros de seguridad se debe brindar un login y password a cada cliente para que sólo los usuarios autorizados puedan acceder a la aplicación. Lo mismo debe ser para el personal de la empresa en sistemas.

8. El sistema debe guardar un histórico de registros (requerimientos y reportes de falla) en una base de datos.

4.4 Requerimientos no funcionales.

Las herramientas que se utilizarán para el desarrollo del sistema serán las siguientes:

1. El sistema se desarrollará sobre Tomcat.
2. Se requiere de Apache para que actúe como servidor para la aplicación.
3. Como plataforma de base de datos se utilizará MySQL.
4. Para el desarrollo de las páginas dinámicas se utilizará tecnología JSP, y servlets con J2EE.
5. Como sistema operativo se utilizará Linux.
6. Por último, se complementará con Macromedia Flash y Fireworks para el diseño de las interfaces visuales.

4.5 Propuesta de solución.

Como solución a la problemática actual que se presenta para la empresa en desarrollo de sistemas, surge la idea de desarrollar un sistema que sea capaz de registrar cada uno de las peticiones solicitadas.

Siendo su principal objetivo controlar y acotar los requerimientos solicitados, es decir, ser una bitácora de eventos a través del tiempo.

El sistema pretende ser un apoyo en la organización del área de operaciones de la empresa agilizando el manejo de información.

4.6 Diagrama General de Casos de Uso.

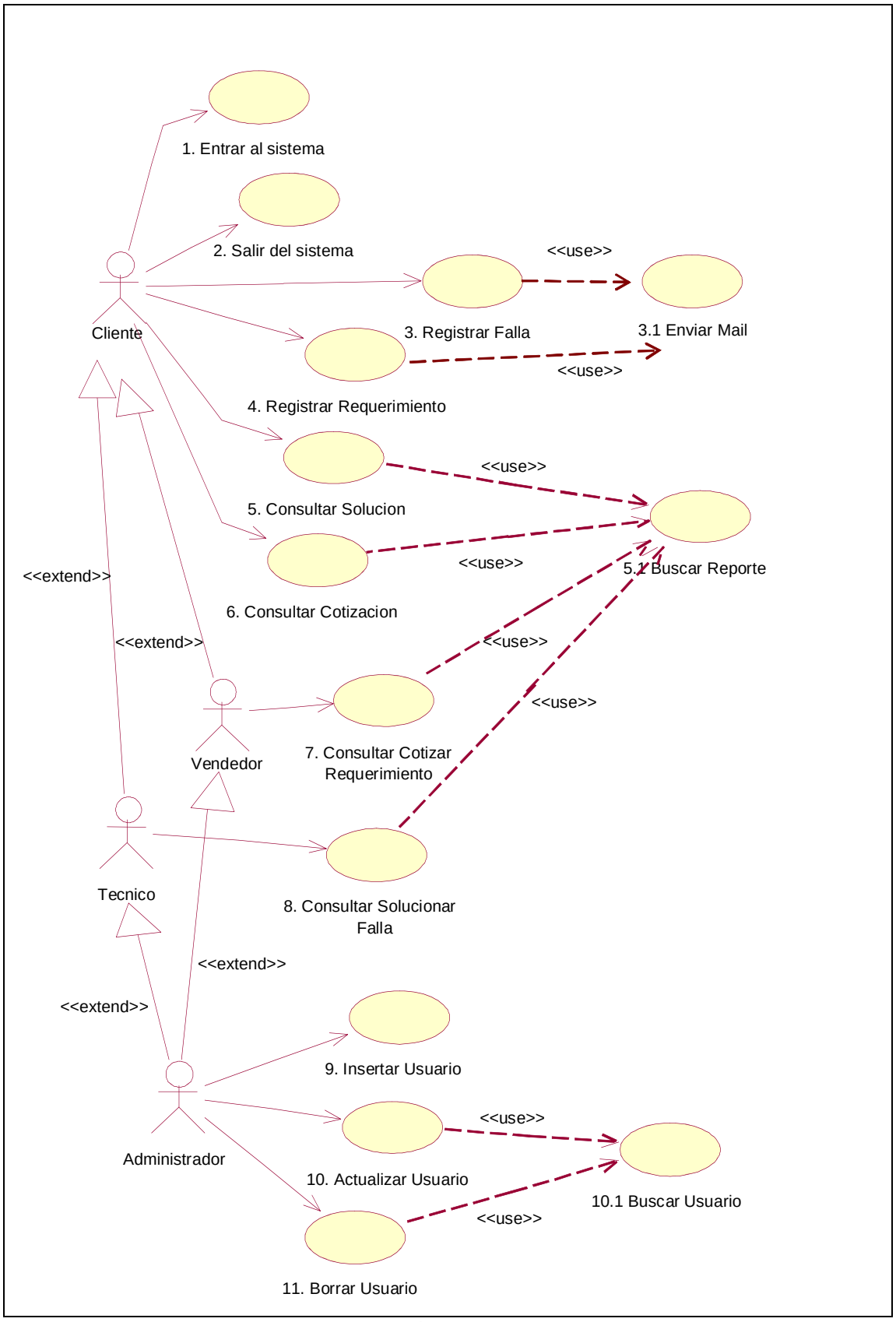


Fig. 4.1

4.6.1 Casos de Uso formato Expandido.

Caso de uso: 1. Entrar al sistema

Formato: Expandido

Actor: Cliente

Descripción: El cliente ingresa su nombre de usuario y password para entrar al sistema.

Precondiciones:

- El cliente debe haber sido dado de alta previamente en el sistema y debe de contar con un nombre de usuario y un password.
- El cliente debe acceder a la página de registro del sistema.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accesa a la página de registro del sistema (inicial).	2	Muestra la página de registro.	
3	Captura el nombre de usuario y password, da aceptar.	4	El sistema muestra la página principal del sistema.	E1

Excepciones:

Id	Nombre	Acción
E1	Si los datos son inválidos.	El sistema indica que la información no es válida y permite nuevamente su captura.

Poscondiciones:

- El cliente se encuentra dentro del sistema.

Caso de uso: 2. Salir del sistema.

Formato: Expandido

Actor: Cliente

Descripción: El cliente sale del sistema.

Precondiciones:
- El cliente debe estar registrado en el sistema.
- El cliente debe encontrarse dentro del sistema.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Selecciona la opción de salida del sistema.	2	Se cierra la aplicación.	

Poscondiciones:
- El cliente se encuentra fuera del sistema.

Caso de uso: 3. Registrar Falla.

Formato: Expandido

Actor: Cliente

Descripción: El cliente registra un reporte de falla.

Precondiciones:
- El cliente debe estar registrado en el sistema.
- El cliente debe haber entrado previamente al sistema.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de registrar falla.	2	Muestra la página de registrar falla.	
3	Captura los datos necesarios y da guardar.	4	El sistema indica que se han guardado los datos, envía un mail de notificación y envía a la página principal.	E1

Excepciones:

Id	Nombre	Acción
E1	Si los datos son inválidos.	El sistema indica que la información no es válida y permite nuevamente su captura.

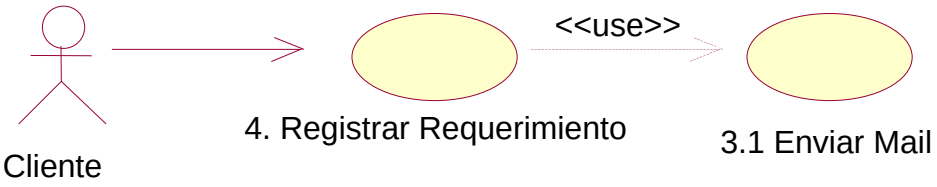
Poscondiciones:

- El cliente se encuentra en la pantalla de inicio.

Caso de uso: 4. Registrar Requerimiento.

Formato: Expandido

Actor: Cliente



Descripción: El cliente registra un reporte de requerimiento.

Precondiciones:

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

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de registrar requerimiento.	2	Muestra la página de registrar requerimiento.	
3	Captura los datos necesarios y da guardar.	4	El sistema indica que se han guardado los datos, envía un mail de notificación y envía a la página principal.	E1

Excepciones:

Id	Nombre	Acción
E1	Si los datos son inválidos.	El sistema indica que la información no es válida y permite nuevamente su captura.

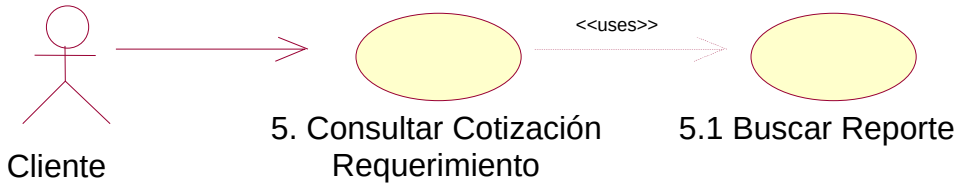
Poscondiciones:

- El cliente se encuentra en la pantalla de inicio.

Caso de uso: 5. Consultar Solución

Formato: Expandido

Actor: Cliente



Descripción: El cliente busca un determinado reporte de falla y consulta la solución proporcionada por parte de un técnico.

Precondiciones:

- El cliente debe estar registrado en el sistema.
- El cliente debe haber entrado previamente al sistema.
- Debe haber sido solucionado un reporte de falla con anterioridad.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de búsqueda de reportes.	2	Muestra la página de búsqueda de reportes.	
3	Busca por identificador de reporte o selecciona la opción de todos los reportes.	4	Muestra una lista de reportes (fallas) que coincidieron con los parámetros de búsqueda.	E1
5	Selecciona el reporte que va a ser consultado.	6	Muestra la página ver solución a falla con los datos del reporte seleccionado.	

Excepciones:

Id	Nombre	Acción
E1	Si no existe ninguna coincidencia en la búsqueda.	El sistema indica que no ha encontrado nada con esos parámetros de búsqueda y permite una nueva búsqueda.

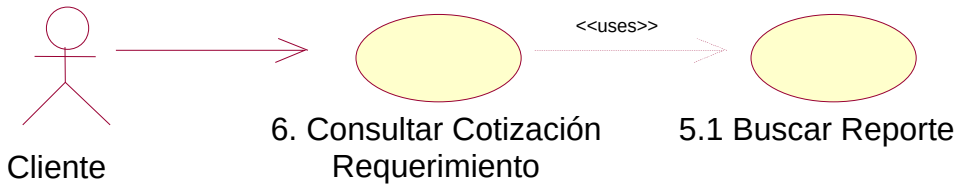
Poscondiciones:

- No hay poscondiciones para este caso de uso.

Caso de uso: 6. Consultar Cotización.

Formato: Expandido

Actor: Cliente



Descripción: El cliente busca un determinado reporte de requerimiento y consulta la cotización proporcionada por parte de un vendedor.

Precondiciones:

- El cliente debe estar registrado en el sistema.
- El cliente debe haber entrado previamente al sistema.
- Debe haber sido respondido un reporte de cotización con anterioridad.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de búsqueda de reportes.	2	Muestra la página de búsqueda de reportes.	
3	Busca por identificador de reporte o selecciona la opción de todos los reportes.	4	Muestra una lista de reportes (requerimientos) que coincidieron con los parámetros de búsqueda.	E1
5	Selecciona el reporte que va a ser consultado.	6	Muestra la página ver cotización a requerimiento con los datos del requerimiento capturado y la cotización al mismo.	

Excepciones:

Id	Nombre	Acción
E1	Si no existe ninguna coincidencia en la búsqueda.	El sistema indica que no ha encontrado nada con esos parámetros de búsqueda y permite una nueva búsqueda.

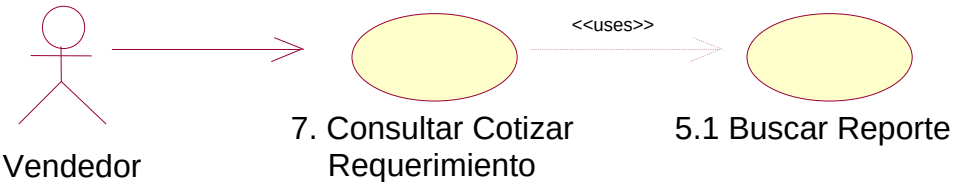
Poscondiciones:

- No hay poscondiciones para este caso de uso.

Caso de uso: 7. Consultar Cotizar Requerimiento.

Formato: Expandido

Actor: Vendedor



Descripción: El vendedor busca un determinado reporte de requerimiento, consulta sus detalles y proporciona una cotización.

Precondiciones:

- El vendedor debe estar registrado en el sistema.
- El vendedor debe haber entrado previamente al sistema.
- Debe haber sido capturado un reporte de requerimiento con anterioridad.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de búsqueda de reportes.	2	Muestra la página de búsqueda de reportes.	
3	Busca por identificador de reporte o selecciona la opción de reportes del día.	4	Muestra una lista de reportes (requerimientos) que coincidieron con los parámetros de búsqueda.	E1
5	Selecciona el reporte que va a ser respondido.	6	Muestra la página de consulta y cotización de requerimientos con los datos del reporte seleccionado.	
7	Captura los datos necesarios para dar respuesta a la cotización y guarda la información.	8	El sistema indica que se han guardado los datos capturados y envía a la página principal.	E2

Excepciones:

Id	Nombre	Acción
E1	Si no existe ninguna coincidencia en la búsqueda.	El sistema indica que no ha encontrado nada con esos parámetros de búsqueda y permite una nueva búsqueda.
E2	Si los datos son inválidos.	El sistema indica que la información no es válida y permite nuevamente su captura.

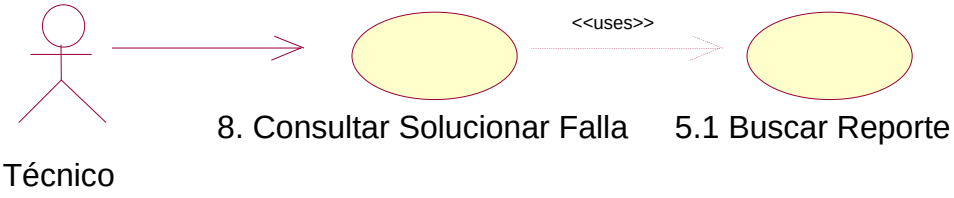
Poscondiciones:

- El vendedor se encuentra en la pantalla de inicio.

Caso de uso: 8. Consultar Solucionar Falla.

Formato: Expandido

Actor: Técnico



Descripción: El técnico busca un determinado reporte de falla y consulta la solución proporcionada por el sistema.

Precondiciones:

- El técnico debe estar registrado en el sistema.
- El técnico debe haber entrado previamente al sistema.
- Debe haber sido respondido un reporte de falla con anterioridad.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de búsqueda de reportes.	2	Muestra la página de búsqueda de reportes.	
3	Busca por identificador de reporte o selecciona la opción de reportes del día.	4	Muestra una lista de reportes (fallas) que coincidieron con los parámetros de búsqueda.	E1
5	Selecciona el reporte que va a ser respondido.	6	Muestra la página de consulta y solución de fallas con los datos del reporte seleccionado.	

7	Captura los datos necesarios para dar solución a la falla y guarda la información.	8	El sistema indica que se han guardado los datos capturados y envía a la página principal.	E2

Excepciones:

Id	Nombre	Acción
E1	Si no existe ninguna coincidencia en la búsqueda.	El sistema indica que no ha encontrado nada con esos parámetros de búsqueda y permite una nueva búsqueda.
E2	Si los datos son inválidos.	El sistema indica que la información no es válida y permite nuevamente su captura.

Poscondiciones:

- No hay poscondiciones para este caso de uso.

Caso de uso: 9. Insertar Usuario

Formato: Expandido

Actor: Administrador

Descripción: El administrador inserta un usuario nuevo en el Sistema IRM.

Precondiciones:

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

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de agregar usuarios.	2	Muestra la página de agregar usuarios.	
3	Captura los datos necesarios para insertar un usuario y da guardar.	4	El sistema informa que los datos han sido capturados exitosamente.	E1

Excepciones:

Id	Nombre	Acción
E1	Si los datos son inválidos.	El sistema indica que la información no es válida y permite nuevamente su captura.

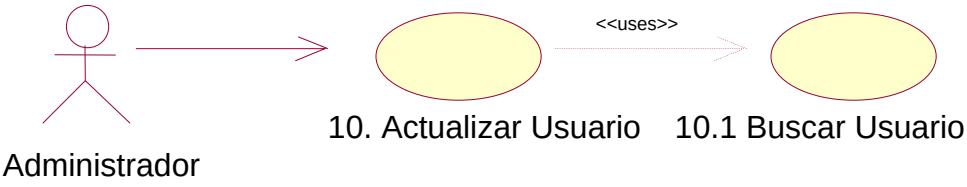
Poscondiciones:

- Existe un nuevo registro de usuario que puede acceder al sistema.

Caso de uso: 10. Actualizar Usuario

Formato: Expandido

Actor: Administrador



Descripción: El administrador actualiza un usuario del Sistema.

Precondiciones:

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

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de búsqueda de usuarios.	2	Muestra la página de búsqueda de usuarios.	
3	Busca por identificador o por nombre de usuario.	4	Muestra una lista de usuarios que coincidieron con los parámetros de búsqueda.	E1
5	Selecciona el usuario que va a ser Actualizado.	6	Muestra la página de actualización de usuarios con los datos del usuario seleccionado.	
7	Modifica los datos que desea cambiar en el usuario y da modificar.	8	Muestra un mensaje de que se modificó el usuario, envía a la pantalla de inicio	E2

Excepciones:

Id	Nombre	Acción
E1	Si no existe ninguna coincidencia en la búsqueda.	El sistema indica que no ha encontrado nada con esos parámetros de búsqueda y permite una nueva búsqueda.
E2	Si los datos de la modificación para actualizar son inválidos.	El sistema indica que la información no es válida y permite nuevamente su modificación.

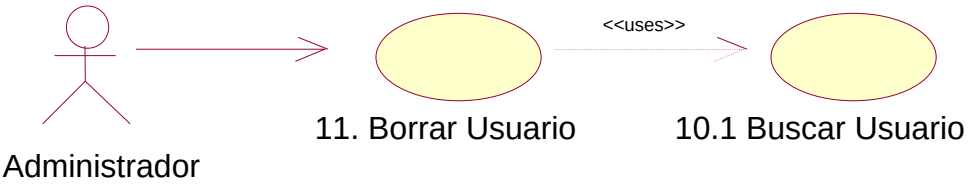
Poscondiciones:

- El administrador se encuentra en la pantalla de inicio.

Caso de uso: 11. Borrar Usuario

Formato: Expandido

Actor: Administrador



Descripción: El administrador borra un usuario del Sistema.

Precondiciones:

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

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Accede a la página de búsqueda de usuarios.	2	Muestra la página de búsqueda de usuarios.	
3	Busca por identificador o por nombre de usuario.	4	Muestra una lista de usuarios que coincidieron con los parámetros de búsqueda.	E1
5	Selecciona el usuario que va a ser eliminado.	6	Muestra un mensaje de que se eliminó el usuario, envía a la pantalla de inicio	

Excepciones:

Id	Nombre	Acción
E1	Si no existe ninguna coincidencia en la búsqueda.	El sistema indica que no ha encontrado nada con esos parámetros de búsqueda y permite una nueva búsqueda.

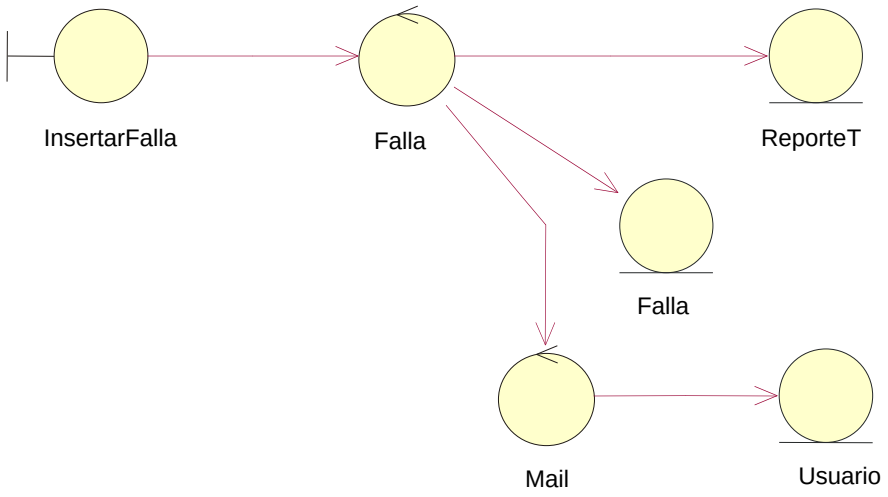
Poscondiciones:

- El administrador se encuentra en la pantalla de inicio.

4.7 Diagramas de análisis.

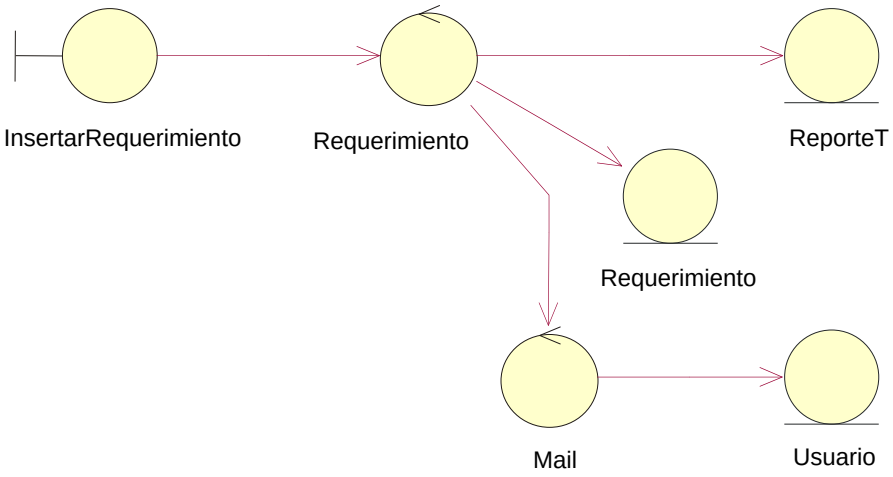
A continuación se muestran los diagramas de análisis que explican el desarrollo de cada caso de uso de acuerdo a las interfaces, estereotipos de control y de entidad.

4.7.1 Registrar Falla.



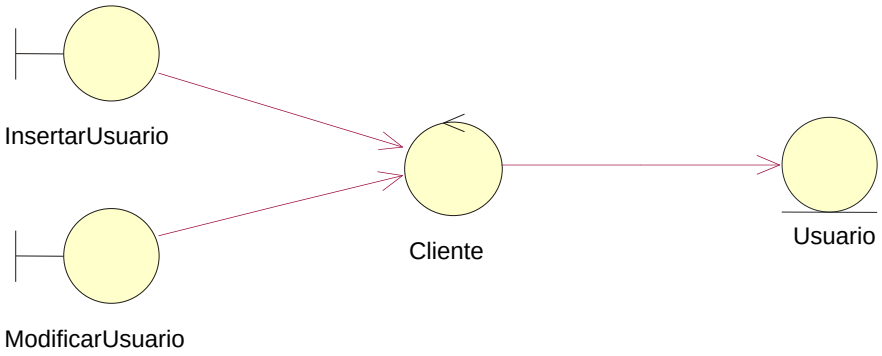
Para el registro de una falla será necesaria una interfaz para la captura de datos la cual llevará el nombre de insertarFalla, esta interfaz enviará la información capturada a una clase de control llamada Falla, que se encargará de guardar la información en las tablas correspondientes (ReporteT y Falla) y enviar un mail de notificación de que se ha registrado una falla nueva; para esto se apoya de la clase Mail que a su vez utiliza, una tabla llamada Usuario, que contiene los datos necesarios para el envío del correo.

4.7.2 Registrar Requerimiento.



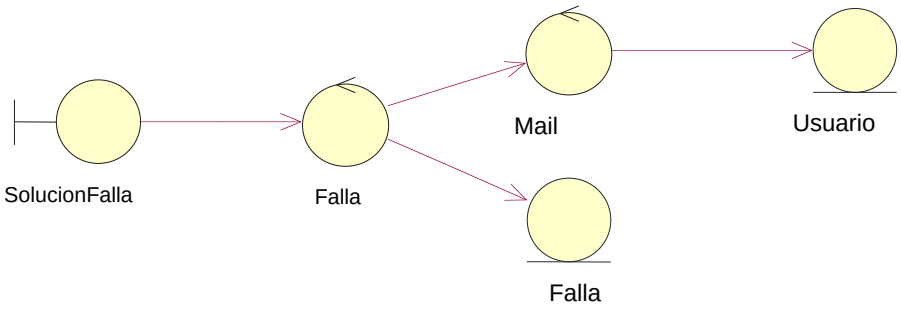
El proceso para el registro de un requerimiento es similar al de registro de falla sólo que la interfaz para la captura de datos llevará el nombre de insertarRequerimiento, y la clase de control Requerimiento guardará la información en las tablas específicas (en este caso ReporteT y Requerimiento) para el registro de requerimientos. De igual forma, se envía un correo de notificación.

4.7.3 Insertar y Modificar Usuario.



El proceso de inserción y modificación de usuarios es similar, ambos requieren de una interfaz para captura de datos, insertarUsuario y modificarUsuario respectivamente. Posteriormente, se utiliza la clase de control Cliente que se encarga de guardar información nueva o modificada en la tabla Usuario. Cabe destacar que para modificar un usuario, además, es necesario hacer una búsqueda previa.

4.7.4 Solucionar Falla.

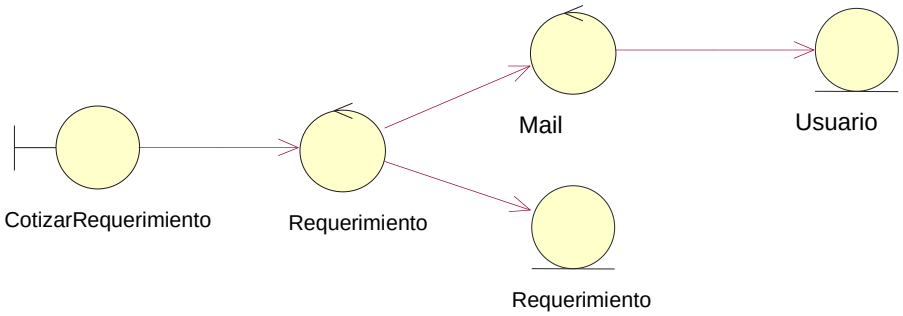


Para solucionar una falla se requiere una interfaz para la captura de la solución que se apoya en una clase de control con el nombre Falla que guarda la solución de la falla en la tabla de falla, como se puede observar, para esta acción no se utiliza la tabla ReporteT, esto es debido a que los datos de solución sólo impactan en la tabla Falla.

De igual modo que cuando se registra una falla, se enviará un correo informativo de que ya existe una solución para una determinada falla.

Cabe destacar que para poder solucionar una falla, es necesario hacer una búsqueda del reporte correspondiente.

4.7.5 Cotizar Requerimiento.



El proceso para cotizar un requerimiento es muy similar al de solucionar falla, solo que para capturar la cotización la clase en que se apoya lleva el nombre Requerimiento. También este proceso enviará un correo de que ya existe cotización para un requerimiento.

5 Diseño del Sistema.

El diseño del sistema será realizado a través de dos tipos de estructuras, que son los diagramas de clases y de secuencia. Con ellos se pretende brindar un panorama de cómo será la estructura de la aplicación, así como los eventos que se presenten en el tiempo de ejecución.

5.1 Diagrama de Clases a Bajo Nivel.

En el siguiente diagrama de Clases a Bajo Nivel se muestran las clases a utilizar y la relación que tienen entre sí, aunque no se especifican los métodos ni los atributos, se puede ver que Clase Cliente, utiliza la clase Mail, y que ReporteT, a su vez contiene, Falla y Requerimiento; ReporteT también utiliza a la clase Mail, así mismo podemos notar que la Clase Cliente utiliza a la clase ReporteT. Por último, la clase Usuario también utiliza a la clase Mail para el envío de correos electrónicos.

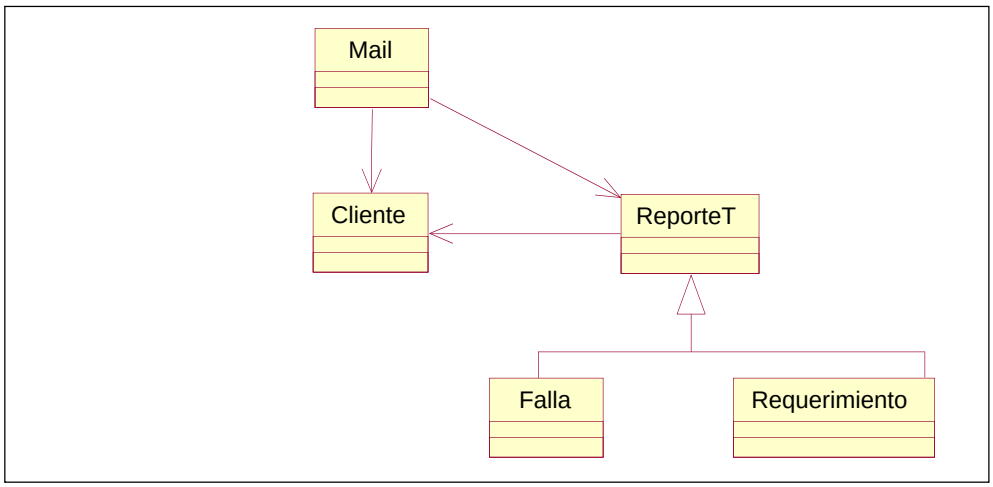


Fig. 5.1 Diagrama de clases de bajo nivel

5.2 Diagrama de Clases a Alto nivel.

En el diagrama de Clases a Alto Nivel que se muestra a continuación podemos observar que ya están integrados los métodos y atributos de cada clase, así como la relación que existe entre ellas pero más a detalle que con el diagrama de Clases a Bajo Nivel. Como se puede observar la clase ReporteT hereda métodos y atributos de Falla y Requerimiento.

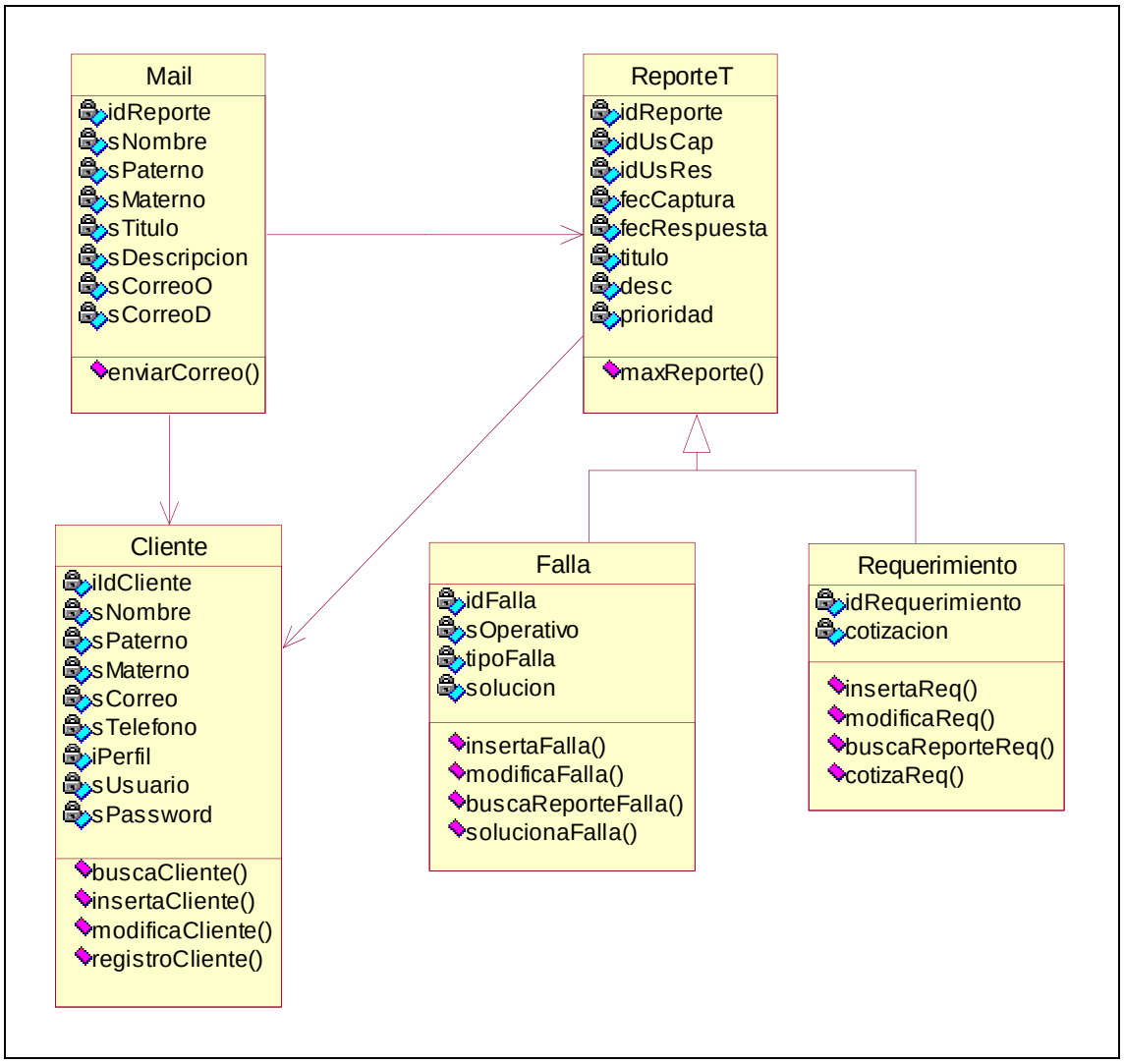


Fig. 5.2 Diagrama de Clases.

Para este diagrama no se incluyeron ninguno de los métodos get y set, debido a que corresponden a uno por atributo y el diagrama se extendería demasiado.

El diagrama es la base para el desarrollo del sistema, basado en él, se realizarán la creación de interfaces y operaciones, además facilitará la explicación de cómo interactuarán los objetos entre sí. En la siguiente sección se verán los diagramas de secuencia los cuales servirán de apoyo para la comprensión de eventos en el tiempo de ejecución.

5.3 ¿Qué es un Diagrama de Secuencia?

En capítulos anteriores se describió brevemente al diagrama de secuencia; a continuación se intenta ahondar más en este tipo de documentación.

El diagrama de secuencia es muy efectivo para modelar interacción entre objetos en un sistema y debe ser modelado para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado desde el punto de vista de negocio, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, así como mensajes intercambiados entre los objetos.

Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se tiene modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces es posible "caminar sobre" esos pasos para descubrir qué objetos son necesarios para poder seguir el flujo de la información.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes intercambiados entre los objetos como vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria.

Los conceptos más importantes relacionados con los diagramas de secuencia son:

- **Línea de vida de un objeto** (*lifeline*): La línea de vida de un objeto representa la vida del objeto durante la interacción. En un diagrama de secuencia un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos a través de la línea principal que denotan la ejecución de métodos (activación). El rectángulo de encabezado contiene el nombre del objeto y el de su clase, en un formato *nombreObjeto : nombreClase*.
- **Activación:** Muestra el período de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto.
- **Mensaje:** El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta.
- **Tiempos de transición:** En un entorno de objetos concurrentes o de demoras en la recepción de mensajes, es útil agregar nombres a los tiempos de salida y llegada de mensajes.
- **Caminos alternativos de ejecución y concurrencia:** En algunos casos sencillos los caminos alternativos pueden expresarse en un diagrama de secuencias alternativas de ejecución. Estas alternativas pueden representar condiciones en la ejecución o diferentes hilos de ejecución (*threads*).
- **Destrucción de un objeto:** Se representa como una *X* al final de la línea de ejecución del objeto.

5.4 Diagramas de Secuencia del sistema.

A continuación se describen los diagramas de secuencia propios del sistema.

5.4.1 Registrar Falla.

Como aclaración para todos los diagramas se podrá observar que las clases tienen un prefijo cls, esto es debido a que existen tablas o actores con el mismo nombre de las clases, por tal motivo se colocó el mencionado prefijo, para diferenciar unos de otros. La única clase que no entra en esta distinción el la clase Mail que conserva su nombre sin modificación.

En este diagrama se puede ver el funcionamiento del caso de uso Registrar Falla, desde donde el cliente se encuentra en la pantalla principal, indica al sistema que desea insertar falla, se abre la pantalla de falla, el cliente captura la información requerida, después valida la información capturada y la clase Falla inserta dentro de la base de datos en las tablas de ReporteT y Falla, ya que el sistema requiere que se inserte en ambas tablas por su arquitectura, después, traspasa la información a la clase Mail, desde ahí se buscan datos del usuario en la tabla Usuario, se regresa la información a Mail y finalmente se envía el correo indicando la falla registrada al usuario deseado.

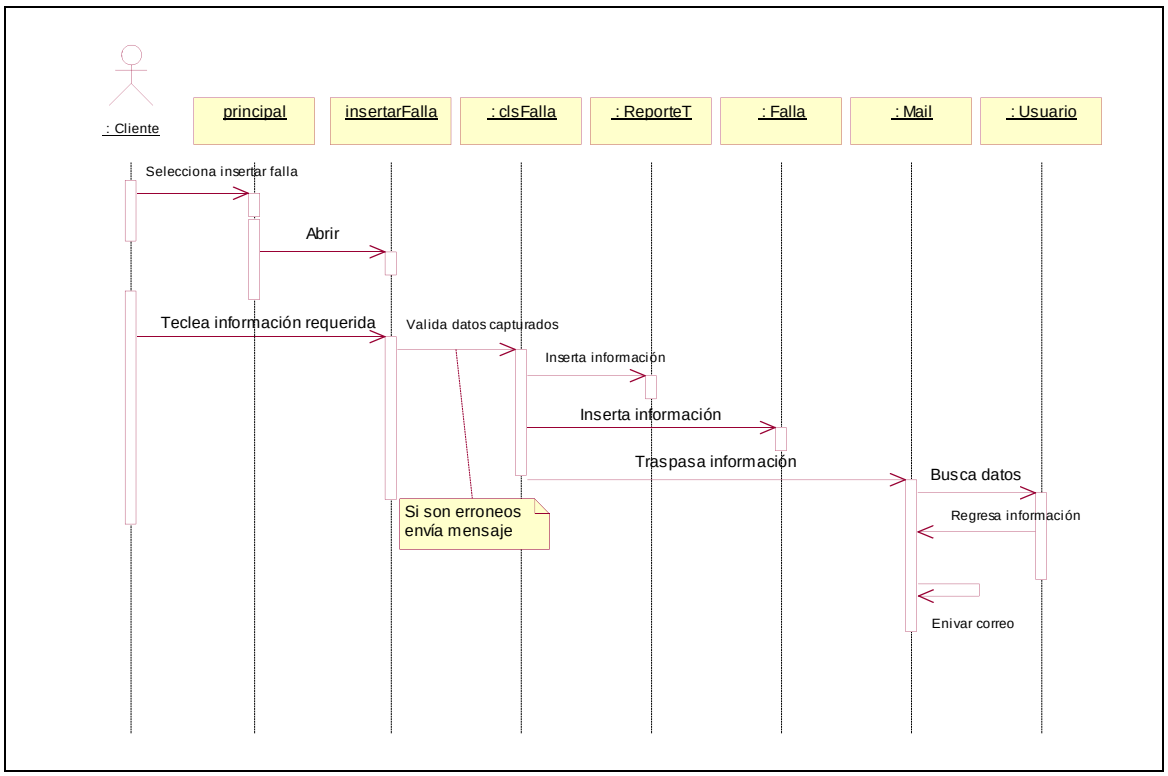


Fig. 5.3 Diagrama de secuencia Registrar Falla.

5.4.2 Registrar Requerimiento.

Este diagrama muestra el funcionamiento del caso de uso Registrar Requerimiento, aquí, el cliente desde la pantalla principal selecciona insertar Requerimiento, una vez abierta la pantalla de requerimiento el cliente captura la información requerida, después valida la información capturada y la clase Requerimiento inserta dentro de la base de datos en las tablas de ReporteT y Requerimiento, a su vez, traspasa la información a Mail, desde donde se buscan los datos de usuario en la tabla Usuario, se regresa la información a Mail y finalmente se envía el correo indicando que el requerimiento ha sido registrado al usuario correspondiente.

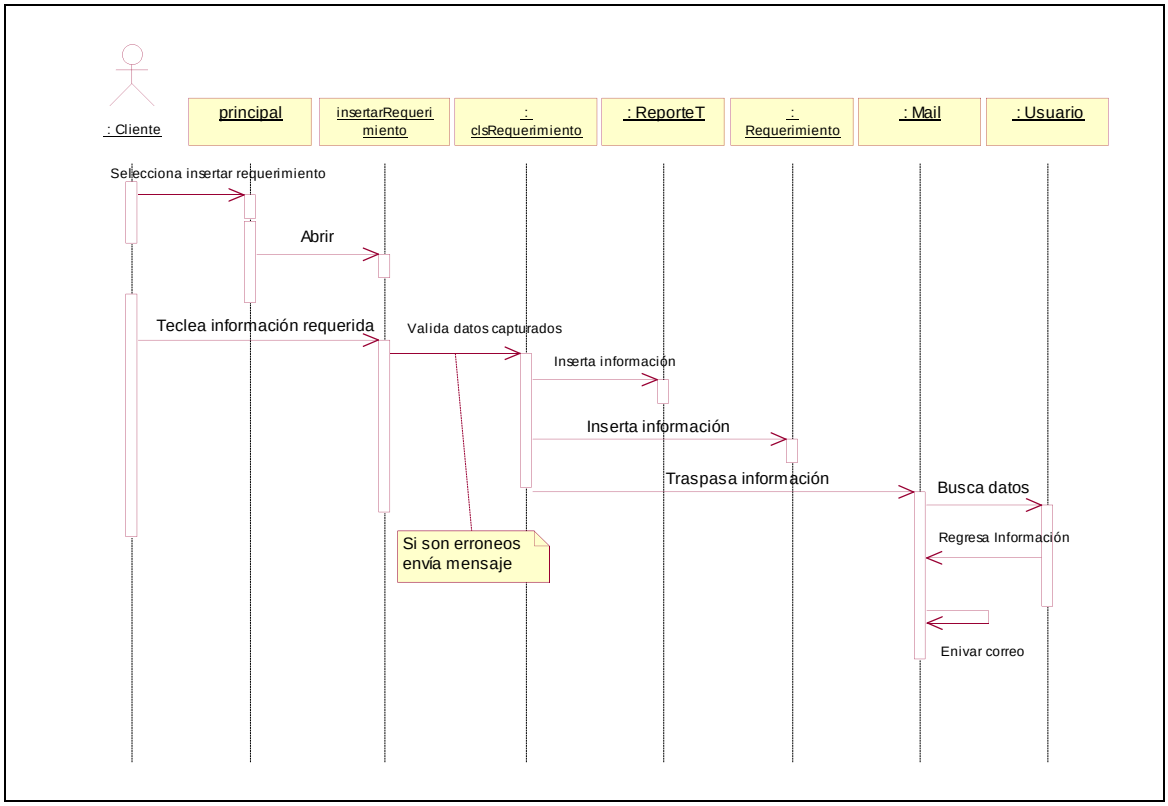


Fig. 5.4 Diagrama de secuencia Registrar Requerimiento.

5.4.3 Consultar Solución.

Aquí, el actor cliente selecciona la opción para buscar un reporte y una vez abierta la pantalla o interfaz teclea la información requerida para realizar la búsqueda, después valida la información y la clase Falla busca el registro en la tabla ReporteT y en la tabla Falla busca la falla, regresa las coincidencias y las muestra, después el cliente selecciona el reporte deseado y finalmente abre la falla.

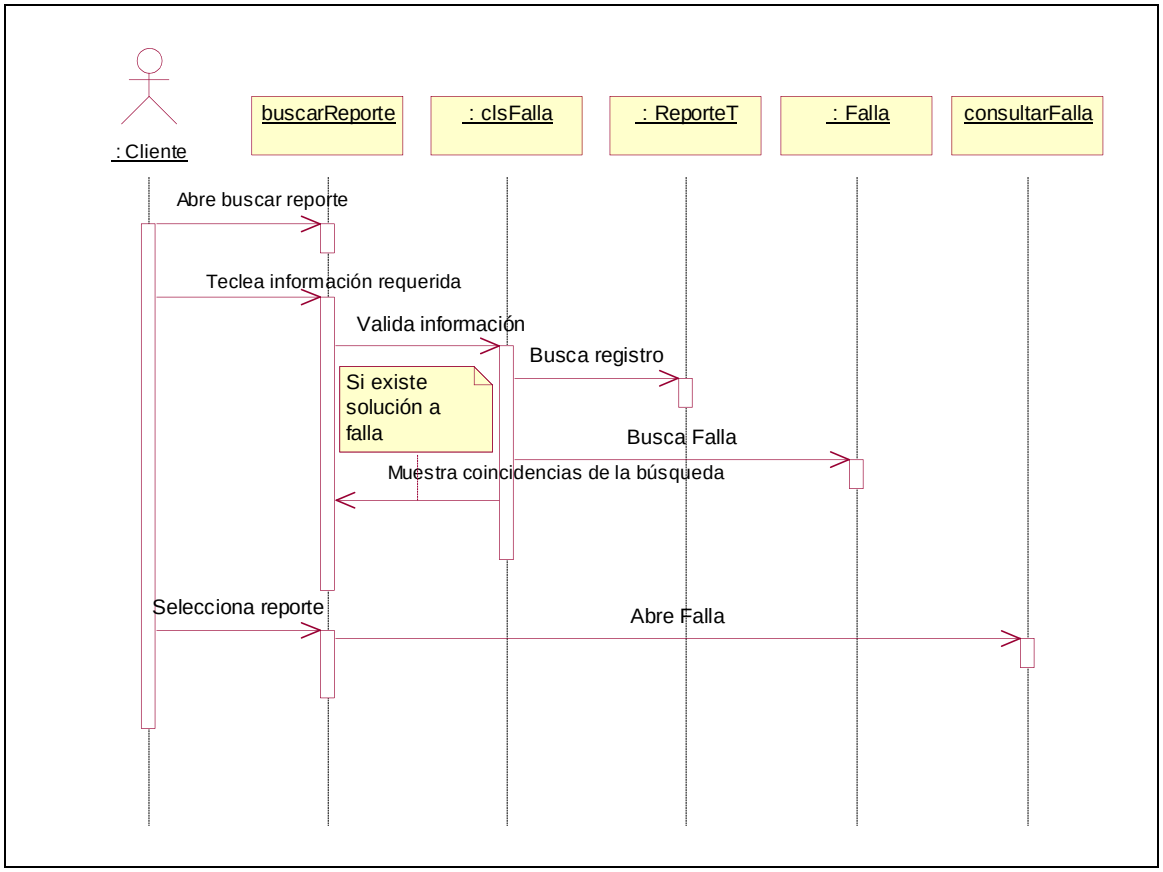


Fig. 5.5 Diagrama de secuencia Consultar Solución.

5.4.4 Consultar Cotización.

El cliente selecciona buscar reporte y una vez abierta la pantalla teclea la información requerida para realizar la búsqueda, ahora para un requerimiento después se valida la información por medio de la clase Requerimiento, esta clase busca el registro en la tabla ReporteT y en la tabla Requerimiento busca el requerimiento, regresa las coincidencias y las muestra, después el cliente selecciona el reporte deseado y finalmente abre el requerimiento.

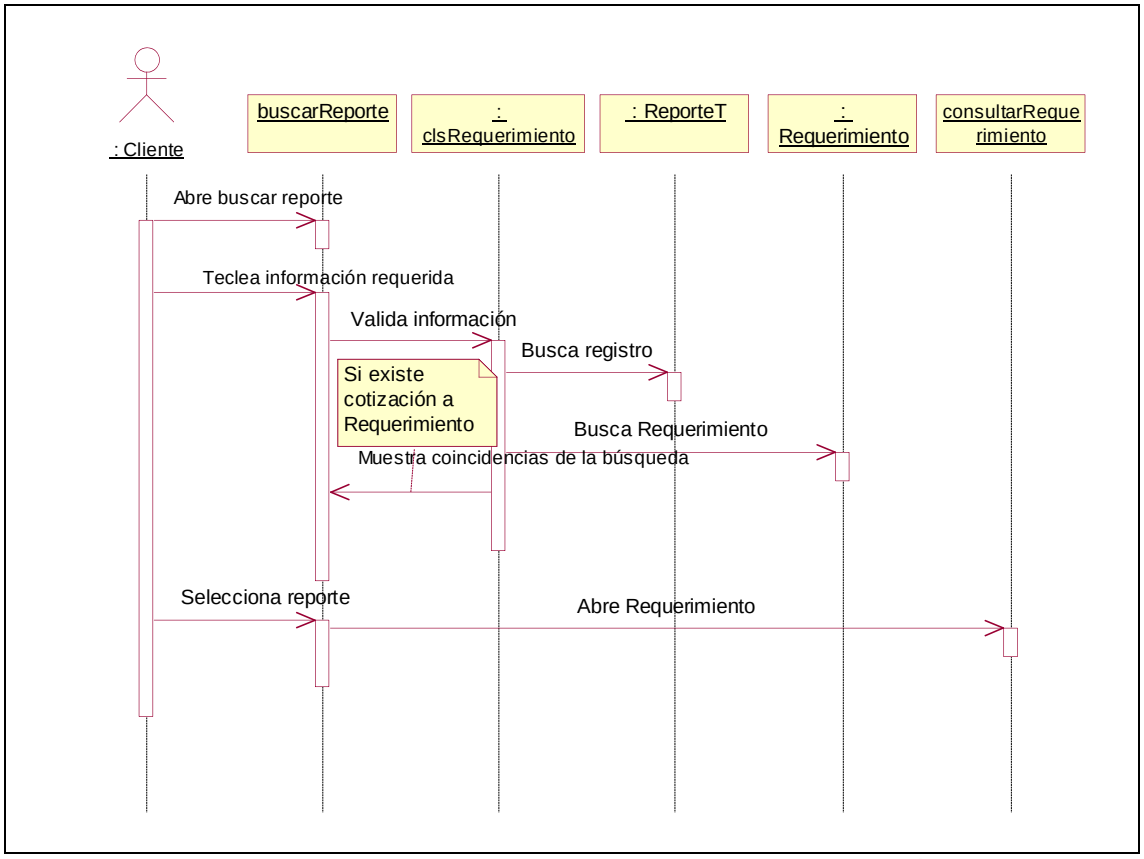


Fig. 5.6 Diagrama de secuencia Consultar Cotización.

5.4.5 Consultar Cotizar Requerimiento.

En el siguiente diagrama el cliente abre la pantalla Buscar Reporte y teclea la información requerida para la búsqueda, valida la información y la clase Requerimiento busca el registro en la tabla ReporteT y el requerimiento en la tabla Requerimiento, regresa las coincidencias encontradas y las muestra en pantalla, el cliente selecciona el reporte deseado y se muestra el requerimiento, después el vendedor teclea la cotización, valida la información y la clase Requerimiento la guarda en la tabla ReporteT y en la tabla Requerimiento.

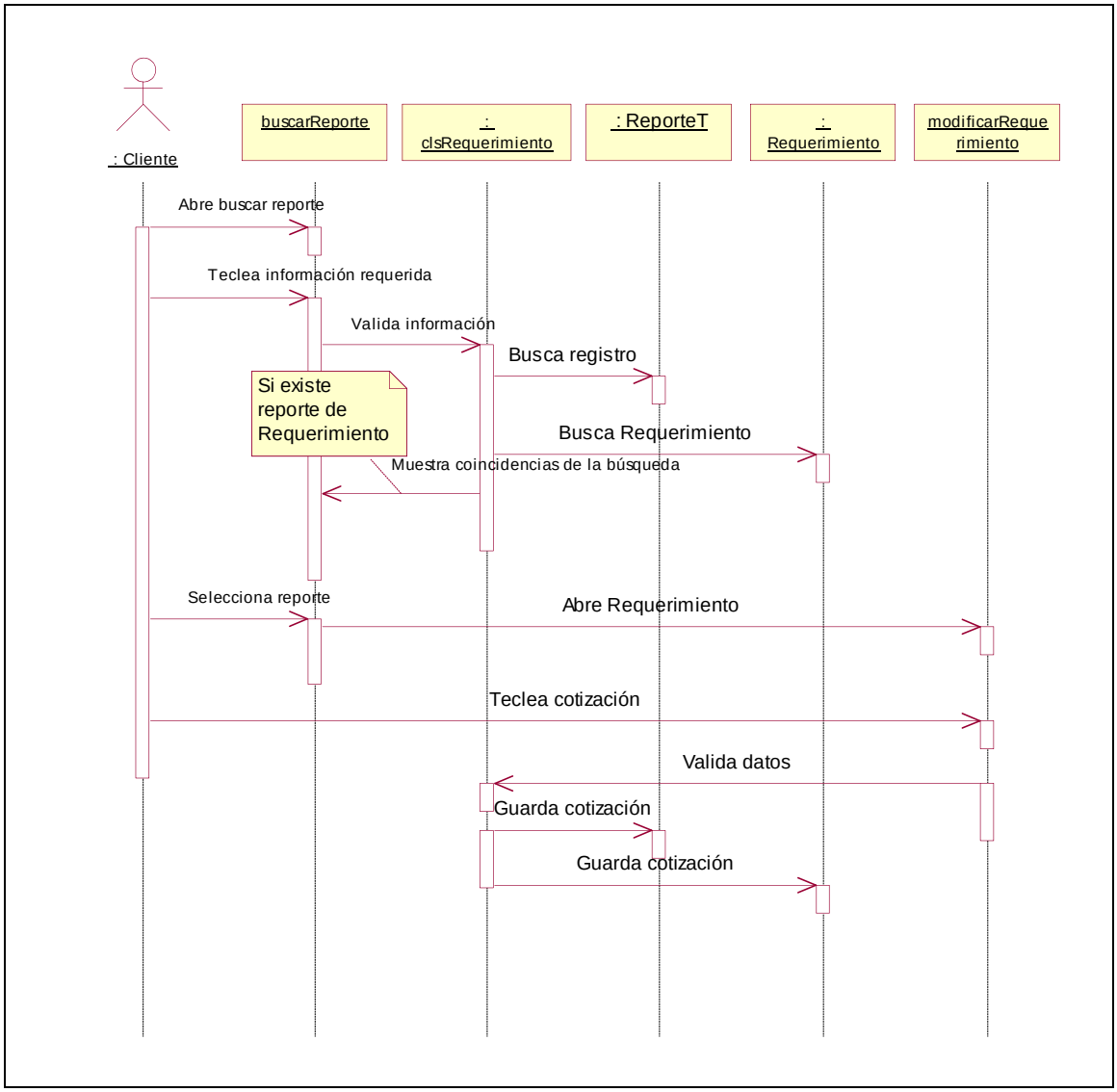


Fig. 5.7 Diagrama de secuencia Cotizar Requerimiento.

5.4.6 Consultar Solucionar Falla.

En este diagrama el técnico selecciona Buscar reporte y teclea la información requerida en la pantalla para realizar la búsqueda, valida la información y la clase Falla busca el registro en la tabla ReporteT y la falla en la tabla Falla, regresa las coincidencias y las muestra en pantalla, el técnico selecciona el reporte deseado y abre la falla, después teclea la información de solución, valida los datos y la clase Falla finalmente los guarda dentro la tabla ReporteT y la tabla Falla.

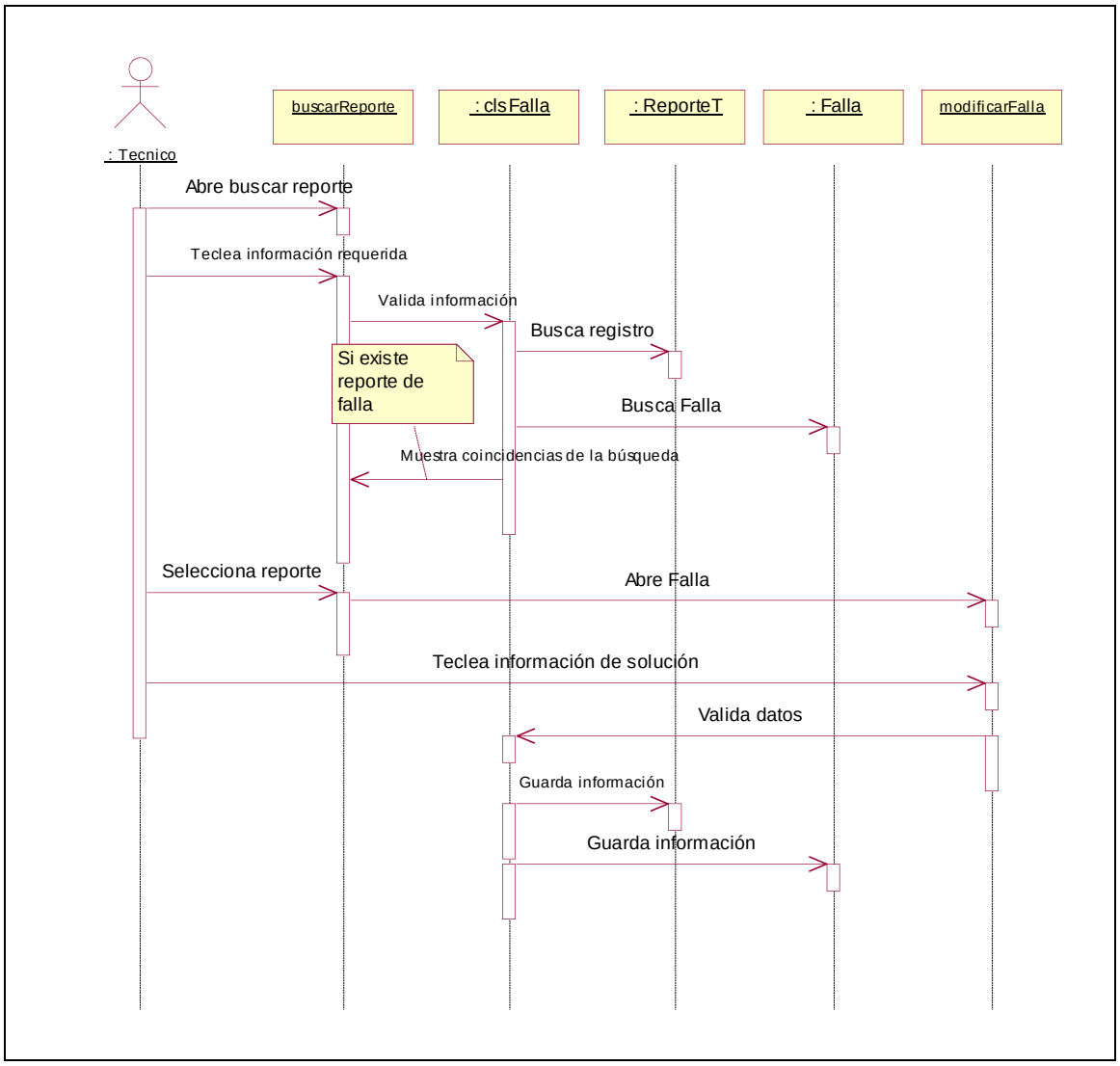


Fig. 5.8 Diagrama de secuencia Solucionar Falla.

5.4.7 Insertar Usuario.

Para insertar un usuario el administrador selecciona desde la pantalla principal insertar usuario y una vez que se ha abierto la pantalla de usuario, el administrador captura la información necesaria para dar de alta un nuevo usuario, posteriormente la clase Cliente valida la información que el actor administrador ha capturado previamente; por último la clase Cliente se encarga de guardar en la tabla Usuario la información.

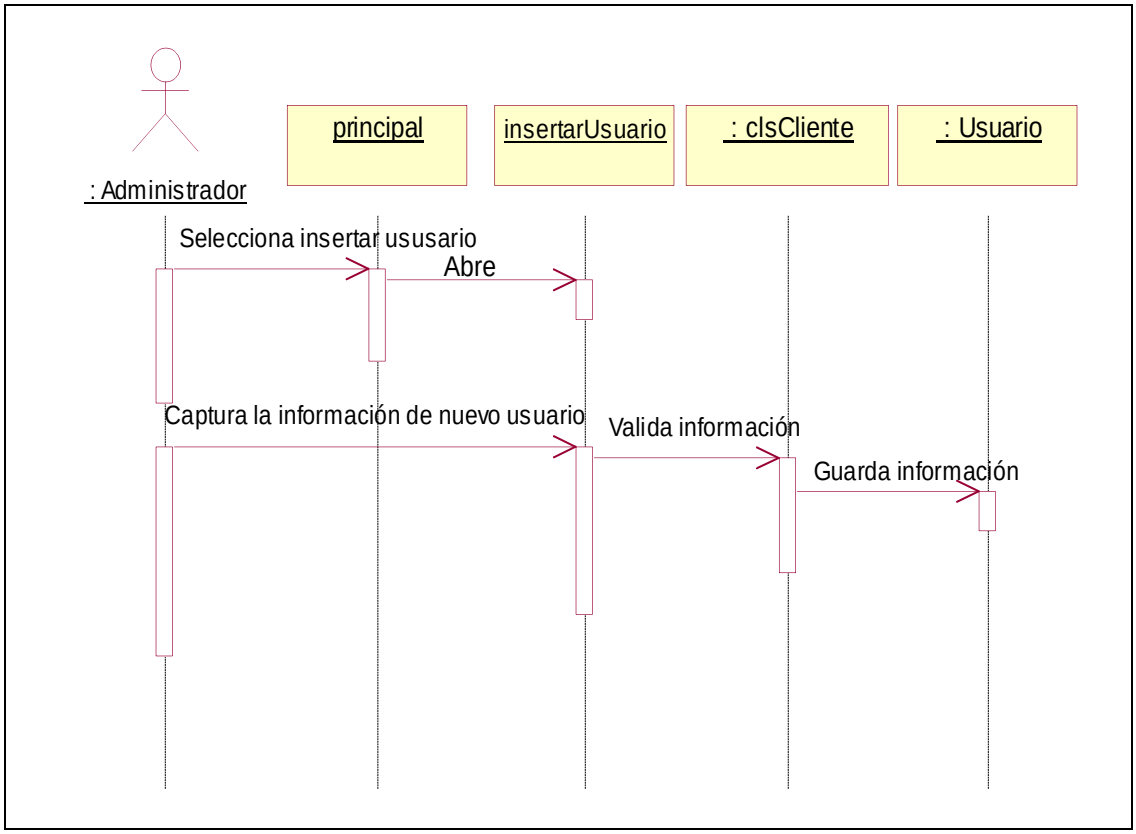


Fig. 5.9 Diagrama de secuencia Insertar Usuario.

5.4.8 Actualizar Usuario.

Para actualizar usuario el diagrama muestra que el administrador desde la pantalla principal selecciona Buscar Usuario, y una vez que se abre dicha pantalla se teclean los parámetros de búsqueda y valida la información, la clase Cliente busca las coincidencias dentro de la tabla Usuario y si existen las muestra en pantalla, después el administrador selecciona y abre al usuario deseado, realiza las modificaciones necesarias, valida los datos y la clase Cliente guarda los cambios dentro de la tabla Usuario.

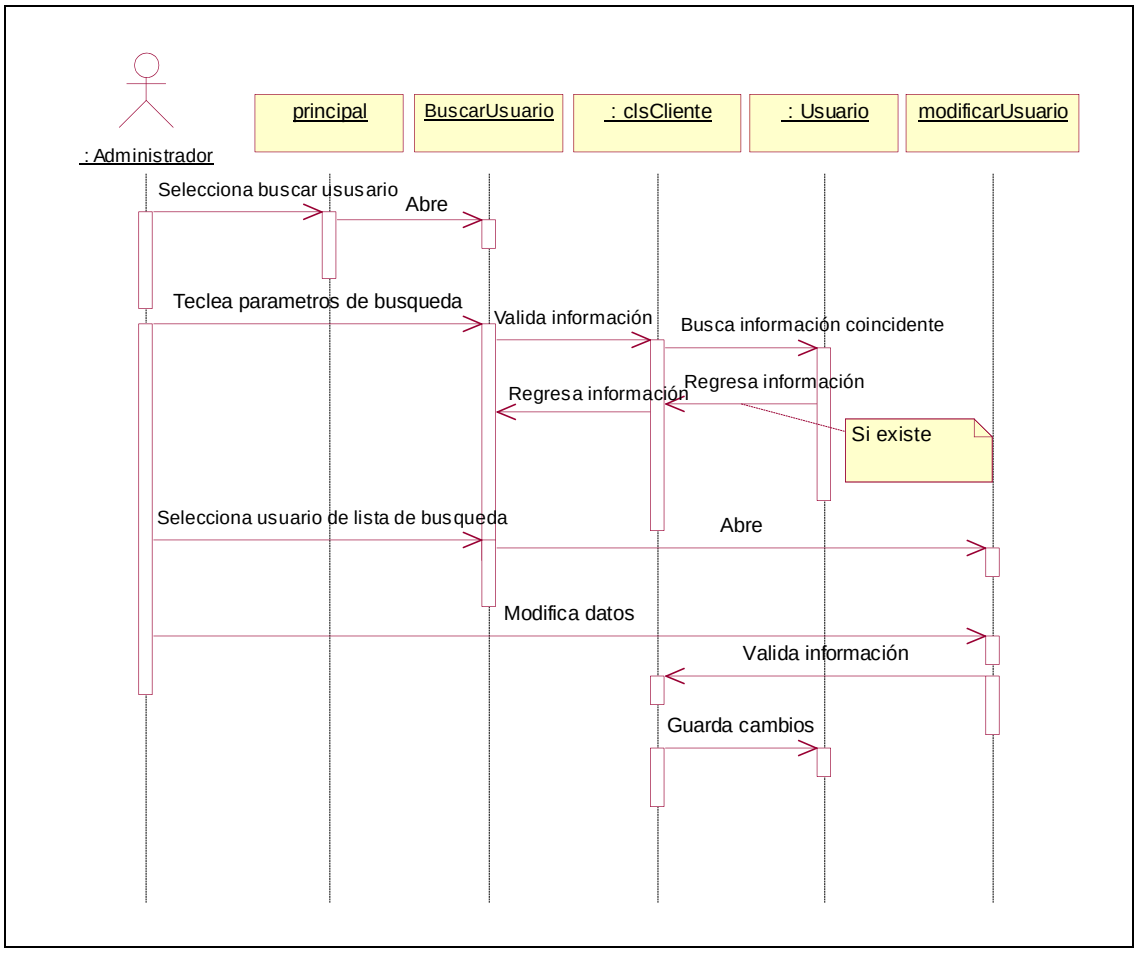


Fig. 5.10 Diagrama de secuencia Modificar Usuario.

5.4.9 Borrar Usuario.

Para buscar un Usuario el administrador abre la pantalla Buscar Usuario y teclea los parámetros de búsqueda y valida los datos, la clase Cliente los busca dentro de la tabla Usuario, si se encuentran coincidencias las regresa y las muestra en pantalla, el administrador selecciona al usuario deseado, lo elimina, y si el administrador confirma que desea eliminarlo, la clase Cliente lo borra del sistema.

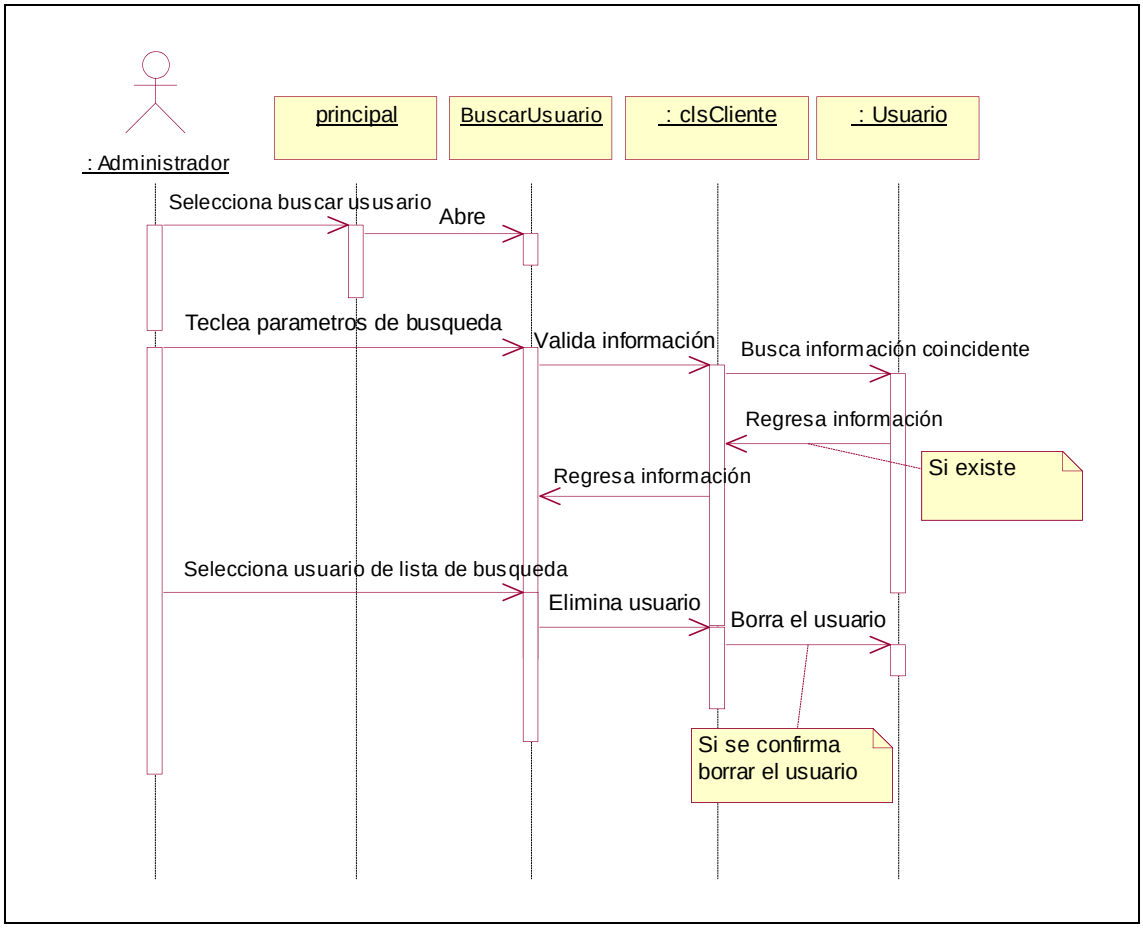


Fig. 5.11 Diagrama de secuencia Borrar Usuario.

Como se pudo observar los diagramas de clases y de secuencia brindan el panorama para que el desarrollo del sistema sea rápido. En el siguiente capítulo se explicarán las interfaces con las que contará el sistema, así como el desarrollo del mismo, en cuanto a código de programación se refiere.

6 Implementación.

En el capítulo anterior se llevó a cabo el análisis y con los diagramas de secuencia se mostró el flujo de la información; ahora en este capítulo se describirá la funcionalidad del sistema pantalla por pantalla y se llevarán a cabo el desarrollo de la aplicación y las pruebas necesarias que constarán de un pequeño prototipo para mostrar al sistema en operación.

6.1 Pantallas de Interfaz.

Las pantallas de la interfaz serán las que interactuarán directamente con el usuario, a continuación se mostrarán figuras que contienen un ejemplo de cómo se verán las pantallas finales.

6.1.1 Pantalla de registro.

Fig. 6.1 Pantalla de registro.

La primera pantalla que mostrará el *sistema para la gestión de fallas y captura de requerimientos vía Web* es el registro para la autenticación de un usuario en el sistema, su funcionalidad principal es dar acceso sólo a los usuarios autorizados para el uso de esta aplicación; esto se realiza por medio de la solicitud de un nombre de usuario y una contraseña únicos por usuario.

6.1.2 Pantalla Principal.

Fig. 6.2 Pantalla principal.

Una vez autorizado un usuario para acceder al sistema se presenta la pantalla principal, fig. 6.2. Esta pantalla solamente da la bienvenida a los usuarios registrados y contiene en su parte derecha un menú para la navegación dentro del sistema.

6.1.3 Registro de falla.

Fig. 6.3 Registro de falla 1.

Esta pantalla se separará en dos imágenes debido a su longitud. Como se puede observar en la figura 6.3 existen algunos datos como el número del reporte que es un consecutivo, la fecha del sistema, la hora del sistema, el nombre del cliente (o usuario) que previamente se ha registrado en el sistema, así como su correo electrónico y número de teléfono. Estos datos son los mismos que presentará la pantalla de registro de requerimiento y son colocados automáticamente por la aplicación, debido a que ya se cuenta con ellos.

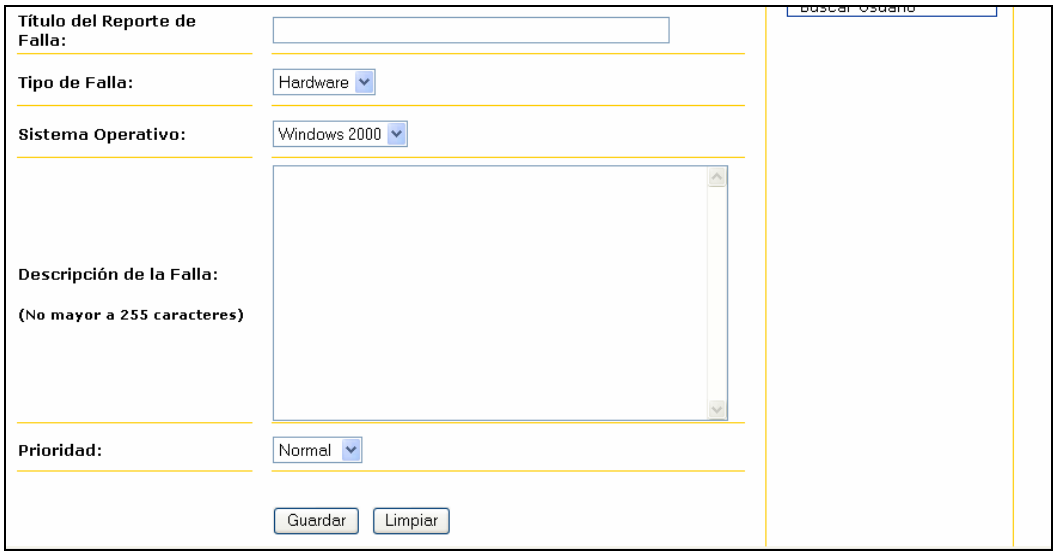


Fig. 6.4 Registro de falla 2.

En la figura 6.4 se muestra la continuación de la pantalla; aquí se puede observar algunos otros datos, sólo que para estos si es necesario que sean especificados por el cliente registrado. Son: título del reporte de falla; tipo de falla, esta puede ser de hardware o de software; el sistema operativo, el cual tiene instalado el cliente en su computadora; la descripción de la falla; aquí se deberá capturar una breve descripción de la falla que afecta al cliente; y por último la prioridad, es decir, la importancia de la falla.

Posteriormente se muestra los botones de guardar y limpiar; el primero sirve para registrar la información capturada por el usuario en la base de datos como un reporte nuevo de falla; a su vez también se enviará un correo electrónico a las personas involucradas para dar solución a la falla (*técnico*), y por último el botón limpiar tiene la funcionalidad de “limpiar” toda la información que haya sido capturada en caso de que se desee introducir datos diferentes.

6.1.4 Registro de requerimiento.

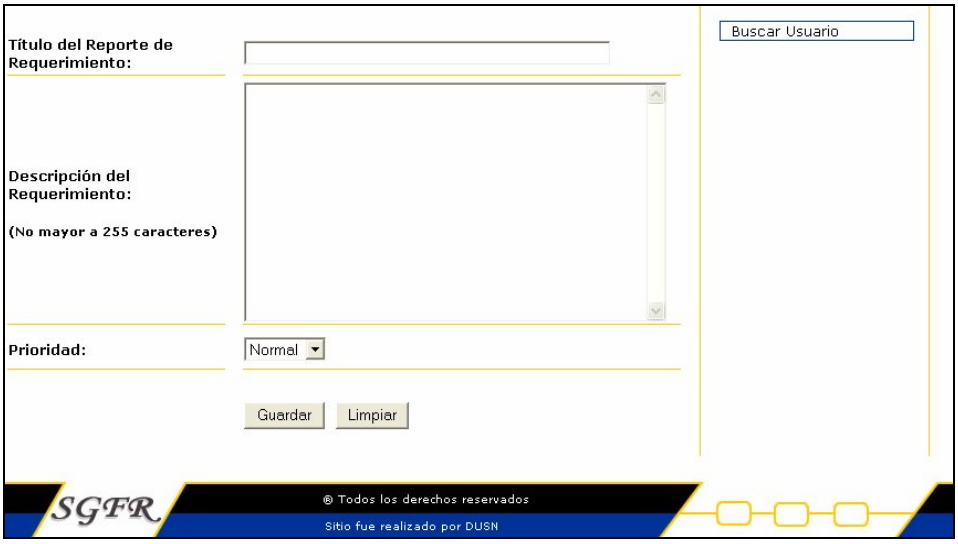
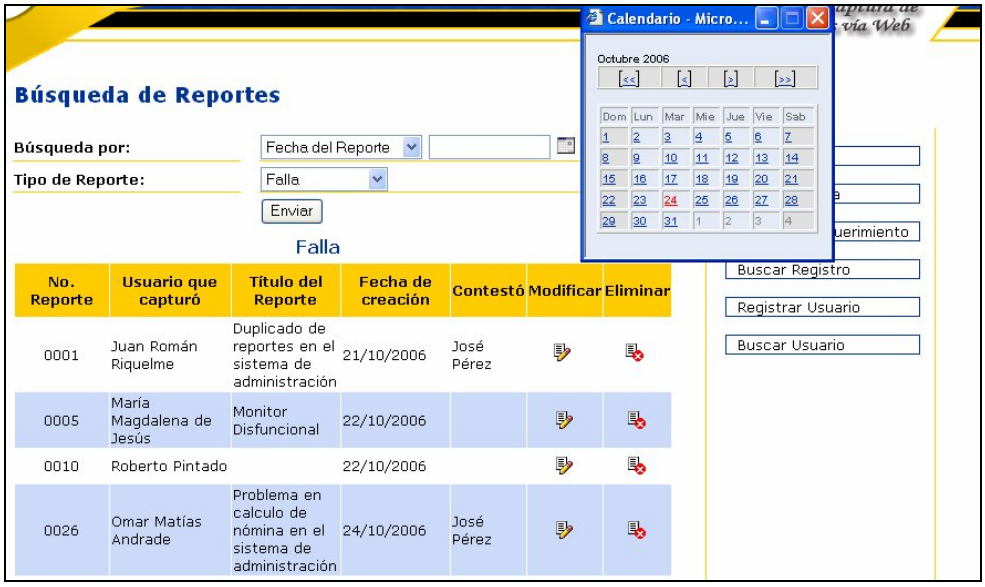


Fig. 6.5 Registro de requerimiento.

Esta pantalla sirve para dar de alta requerimientos, es decir, nuevas funcionalidades en algún sistema, o quizá alguna solicitud de equipo, etc. En la figura 6.3 se muestra los datos dados automáticamente por la aplicación, estos aplican también para el registro de requerimientos, en la figura 6.5 se observan los datos exclusivos de esta pantalla, estos son: título del reporte de requerimiento, descripción del requerimiento y prioridad el reporte; así como los respectivos botones de guardar y limpiar. La pantalla guarda los datos de reporte y a su vez envía un correo informando a los usuarios indicados (*vendedor*) que un nuevo reporte de requerimiento ha sido dado de alta.

6.1.5 Búsqueda de reportes.



No. Reporte	Usuario que capturó	Título del Reporte	Fecha de creación	Contestó	Modificar	Eliminar
0001	Juan Román Riquelme	Duplicado de reportes en el sistema de administración	21/10/2006	José Pérez		
0005	María Magdalena de Jesús	Monitor Disfuncional	22/10/2006			
0010	Roberto Pintado		22/10/2006			
0026	Omar Matías Andrade	Problema en calculo de nómina en el sistema de administración	24/10/2006	José Pérez		

Fig. 6.6 Búsqueda de reporte.

Por medio de esta pantalla los usuarios del sistema pueden encontrar reportes que hayan sido dados de alta con anterioridad, existen dos tipos de búsqueda por número del reporte, este es un dato que la aplicación proporciona automáticamente al registrar un reporte (ya sea de falla o requerimiento), al seleccionar esta opción el usuario deberá introducir el número del reporte que requiera encontrar. El otro tipo de búsqueda es por fecha del reporte, como se puede ver en la figura 6.6, si se da clic en la imagen del calendario aparecerá una ventana con también un calendario para que se seleccionen los rangos de fechas para realizar la búsqueda.

Una vez seleccionado el tipo de búsqueda el usuario deberá seleccionar el tipo de reporte que desea encontrar, es decir, si desea un reporte de tipo falla o uno de tipo requerimiento. Posteriormente deberá dar clic al botón de Enviar.

En la figura 6.6 se muestran los resultados de una búsqueda por fechas y del tipo falla. Los resultados se muestran en una tabla que contiene los siguientes datos: número del reporte, usuario que capturó (dio de alta) el reporte, el título, la fecha de creación, el usuario (técnico o vendedor dependiendo del tipo de reporte) que respondió el reporte en caso de que ya se haya proporcionado

una solución o una cotización para el reporte y dos columnas extras que son las de modificar y eliminar. La columna de modificar sirve para dirigirse al reporte que se desea contestar o modificar, esto se explicará más a detalle en las pantallas de solución de falla y cotización de requerimiento; la columna de eliminar sirve para inactivar el reporte correspondiente, un reporte de falla o requerimiento sólo puede ser eliminado por el usuario que dio de alta el reporte o por el administrador del sistema.

6.1.6 Solución a falla.

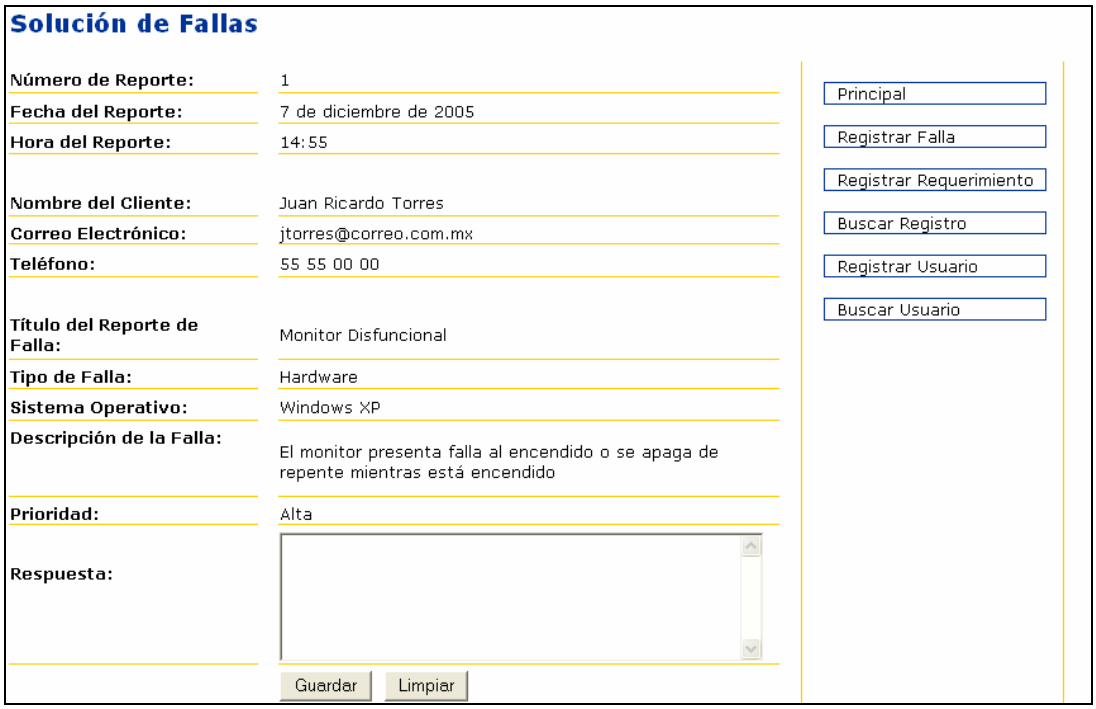


Fig. 6.7 Solución a falla.

Como se mencionó anteriormente en el momento en que se da de alta un reporte de falla se envía un correo electrónico a los técnicos para que se le de solución o para que se especifique de que modo se procederá para dar solución al reporte insertado. Una vez que el técnico recibe el correo, procede a buscar el reporte que es necesario responder, esto se explicó en la sección 6.1.5, después de dar clic en la columna modificar, el sistema pasará a la pantalla que muestra la figura 6.7; como se puede observar el reporte muestra la información previamente capturada por el cliente, en la parte inferior también existe un área de texto en donde se capturará la respuesta proporcionada por el técnico para solucionar la falla, o donde se indicará que acciones procederán para la solución.

Después de proporcionar la solución el técnico deberá dar clic al botón guardar, esto insertará la solución en la base de datos y a su vez enviará un correo electrónico al cliente que dio de alta el reporte para que este pueda revisar la solución a su problema.

6.1.7 Consulta solución a falla.

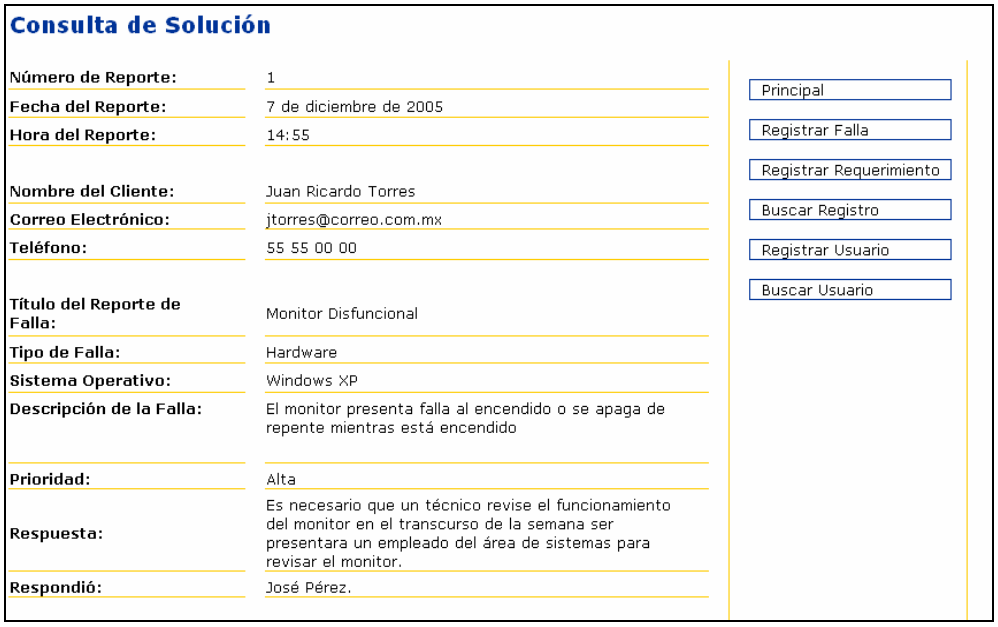


Fig. 6.8 Consulta de solución.

Esta pantalla muestra únicamente la respuesta del técnico a un cliente de un reporte de falla.

6.1.8 Cotización de requerimiento.

Fig. 6.9 Cotización de requerimiento.

La funcionalidad de esta pantalla es similar a la de Solución a Falla, en la figura 6.9 se muestra la información que contiene esta pantalla y se observa que comparte información con la de solución, la única diferencia notable es que únicamente los vendedores pueden acceder a esta sección debido a que su perfil es el único que puede cotizar un requerimiento. Existe un área de texto para la captura de la respuesta y los botones de guardar y limpiar.

De igual forma una vez que el vendedor proporcionó la cotización, se envía un correo al cliente que capturó el reporte.

6.1.9 Consulta cotización de requerimiento.

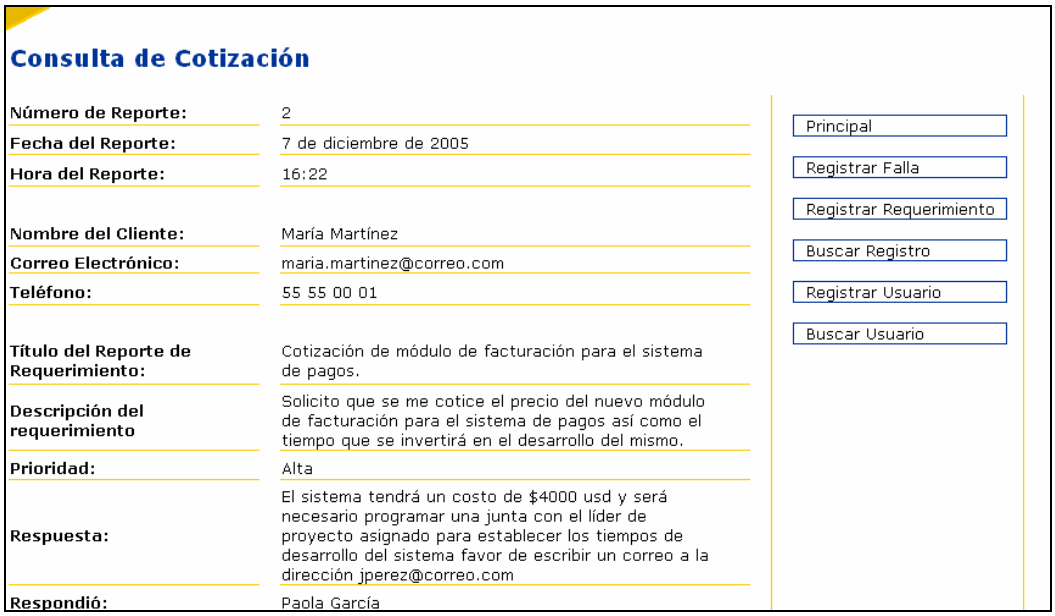


Fig. 6.10 Consulta Cotización.

Esta pantalla muestra la cotización proporcionada por el vendedor al cliente que capturó el reporte de requerimiento. Para acceder a esta pantalla es necesario que el cliente se dirija a la búsqueda de reportes, esto se explicó en la sección 6.1.5

6.1.10 Alta de usuarios.

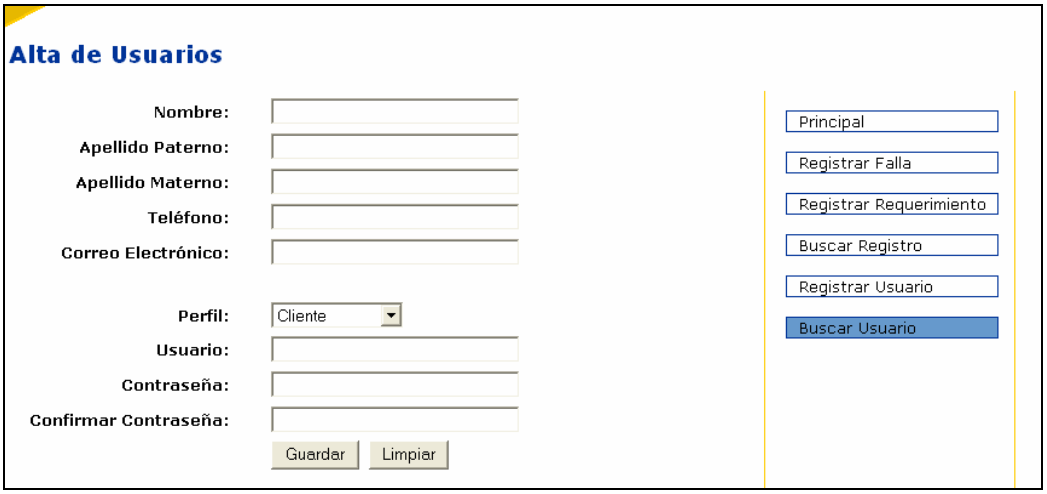


Fig. 6.11 Alta de usuarios.

La pantalla de alta de usuarios tiene como objetivo el dar de alta usuarios para que puedan tener acceso al sistema, el único perfil con las facultades de dar de alta usuarios nuevos es el administrador. En la figura 6.11 se observan los

campos requeridos para poder dar de alta un usuario nuevo y son los siguientes:

Nombre, apellido paterno, apellido materno, teléfono y correo electrónico como datos generales del nuevo usuario que se utilizarán por ejemplo en la pantalla de insertar requerimiento, en donde los datos que proporciona el sistema automáticamente son recogidos aquí; posteriormente se deben definir datos que le servirán a la aplicación para la configuración del usuario dentro del sistema.

El campo perfil es el encargado de distinguir que tipo de usuario será el que se dé de alta, para esto se utiliza un combo en donde se encuentran los 4 perfiles disponibles para este sistema, es decir, cliente, técnico, vendedor y administrador. De acuerdo a la selección del perfil el usuario tendrá permisos o no para ciertas secciones de la aplicación, por ejemplo para dar solución a fallas los únicos que podrán hacerlo será un usuario con perfil de técnico y también el administrador, cabe mencionar que el administrador tiene la facultad de todos los permisos dentro del sistema.

La pantalla de alta de usuarios presenta dos campos de texto que son el "usuario" y la "contraseña", el primero sirve como identificador único para el usuario, por su parte el campo de contraseña es la clave que posee el usuario para poder acceder al sistema, estos datos se utilizan en la sección del registro de usuarios, en donde se determina si un usuario tiene acceso a la aplicación y qué secciones de la aplicación esta autorizado a utilizar.

Por último existe un campo de verificación de contraseña, está es solamente una llave para que se recuerde perfectamente la contraseña del usuario que se esta registrando.

6.1.11 Búsqueda de usuarios.

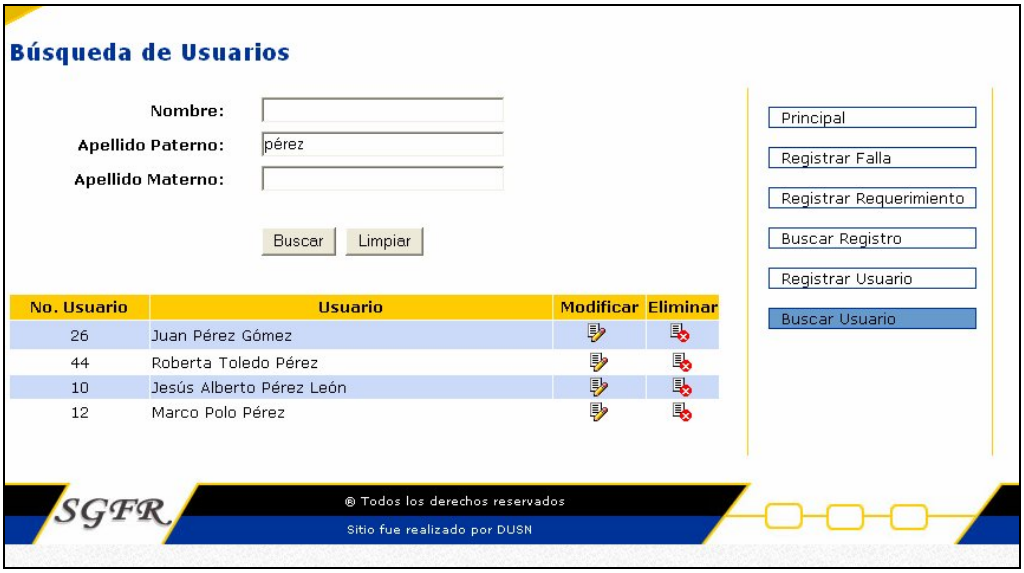


Fig. 6.12 Búsqueda de usuarios.

La figura 6.12 muestra la forma en que es posible la búsqueda de un usuario.

La pantalla tiene la funcionalidad de encontrar usuarios para que puedan ser eliminados o modificados. Para realizar la búsqueda es necesario dar clic al botón de Buscar. Por su parte el botón de Limpiar tiene la finalidad de borrar información previamente capturada en los campos de texto.

Los campos para la búsqueda son: el nombre, el apellido paterno y el materno y de acuerdo a la combinación de los tres campos la búsqueda mostrará las coincidencias pertinentes. Por ejemplo, en la figura 6.12 se observa que se ha realizado una búsqueda por apellido paterno, en el campo de texto se puede ver la palabra *Pérez,* en la parte inferior se muestra una tabla con las coincidencias de la búsqueda con la palabra P*érez.* La tabla cuenta con las siguientes columnas informativas: número de usuario, usuario, modificar y eliminar. Las columnas que vale la pena explicar más a fondo son la de modificar, esta se encarga de conducir al usuario administrador a la pantalla de modificación de usuarios y la columna eliminar que se encarga de dar de baja un usuario; si un usuario es dado de baja ya no tendrá la posibilidad de entrar al sistema.

6.1.12 Actualiza usuario.

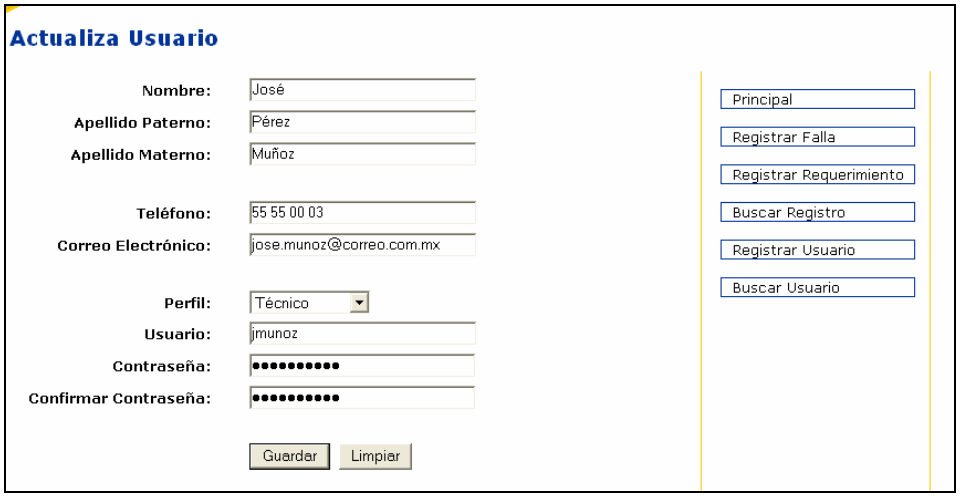


Fig. 6.13 Actualización de usuarios.

Esta pantalla es similar a la de alta de usuarios, sólo que tiene la finalidad de modificar los datos que se capturaron previamente acerca de un usuario. Es importante debido a que si el correo electrónico de determinado usuario es obsoleto no se podrán enviar los correos electrónicos correctamente, también en caso de que algún usuario olvide su contraseña.

6.2 Documentación del desarrollo.

Una vez que se han mostrado las pantallas principales de la aplicación, se procederá a describir el completo funcionamiento de un módulo del sistema.

Se ha elegido el módulo de “Inserción de Fallas” debido a que es una de las dos funcionalidades principales del sistema, también porque el módulo de requerimientos es muy similar al de fallas y a que el módulo de usuarios solo consiste en un simple catálogo.

El sistema cuenta como ya se ha mencionado con 4 perfiles de usuario los cuales son: Cliente, Técnico, Vendedor y Administrador. Dependiendo del perfil de usuario que se registre en la aplicación se habilitarán las secciones a las cuales el usuario tendrá acceso. El módulo de fallas sólo puede ser accedido por un usuario con perfil de Técnico o de Administrador del sistema, es debido a esto que es de primordial importancia el explicar el registro de usuarios.

También es necesario explicar el patrón de diseño MVC debido a que el sistema esta montado y referenciado en esta forma de desarrollo de aplicaciones.

6.2.1 Modelo Vista Controlador (MVC del inglés Model View Controller).

El MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML o en este caso un JSP debido a que se esta utilizando tecnología J2EE; el control es el código que provee de datos dinámicos a la página, el control de esta aplicación se lleva por parte de “Servlets” y el modelo contiene clases de java representativas de la aplicación (como el mensaje de un foro, un miembro registrado, etc.).

- **Modelo**: Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos; por ejemplo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o portes en un carrito de la compra.
- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.
- Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente (como puede ser una base de datos. Para esta aplicación se estará utilizando MySQL) para almacenar los datos. MVC no menciona específicamente esta capa de acceso a datos.

Es común pensar que una aplicación tiene tres capas principales: presentación (IU), dominio, y acceso a datos. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre Vista/Controlador es menos clara.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de alguna forma (por ejemplo, el usuario pulsa un botón, sigue un enlace o realiza una petición)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. Este gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback. (Un callback es la llamada que hace por ejemplo un formaluario HTML por medio de un método, ya sea post o get hacia su action, en este caso hacia un servlet que realiza la petición de información).
3. El controlador (servlet) accede al modelo (clase java), actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta in dirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. *Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.*
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

6.2.2 Registro de Usuarios.

Este módulo se encarga de dar privilegios al usuario que se esté registrando en la aplicación. Estos privilegios se obtienen de acuerdo al perfil de usuario con el que cuenta el individuo. Como ya se ha mencionado, el sistema cuenta con 4 perfiles y a estos se les ha asignado un número identificador único, en la tabla 6.1 se muestran las asignaciones de identificadores por perfil de usuario.

Identificador	Perfil
1	Cliente
2	Vendedor
3	Técnico
4	Administrador

Tabla 6.1 Perfiles de usuario.

El identificador que muestra la tabla anterior sirve para delimitar los permisos de acceso que tiene cada usuario, esto se mostrará a detalle más adelante.

Una vez mostrados los perfiles se procederá a explicar el funcionamiento y las partes que constituyen el módulo de registro.

Partes que integran el módulo.

Este módulo se integra de tres partes las cuales corresponden al patrón MVC, una vista que es un archivo JSP que tiene la función de interfaz entre el usuario real y la aplicación este archivo tiene el nombre de index.jsp; la segunda parte es un servlet llamado registro.java, el cual tiene la función de puente entre el modelo y la vista, por último tenemos al modelo que es una clase java llamada Cliente.java, la cual se encarga de procesar toda la información.

A continuación se explicarán detalladamente cada una de las partes que integran el módulo de registro.

La parte de la vista, es decir, el archivo index.jsp, tiene la función de una interfaz hacia el usuario, a continuación se muestra su código en la tabla 6.2.

```
<%@page contentType=”text/html”%>
<%@page pageEncoding=”UTF-8”%>
<!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>:: S G R F ::</title>
   </head>
<body topmargin=”0” leftmargin=”0” background=”vista/imagenes/backGrande.png”>
   <p> </p>
<table width=”451” border=”0” align=”center” cellpadding=”0” cellspacing=”0”>
<tr>
   <td width=”92”><img src=”vista/imagenes/spacer.gif” width=”92” height=”1” border=”0” alt=””></td>
   <td width=”53”><img src=”vista/imagenes/spacer.gif” width=”53” height=”1” border=”0” alt=””></td>
   <td width=”159”><img src=”vista/imagenes/spacer.gif” width=”159” height=”1” border=”0” alt=””></td>
   <td width=”85”><img src=”vista/imagenes/spacer.gif” width=”85” height=”1” border=”0” alt=””></td>
   <td width=”61”><img src=”vista/imagenes/spacer.gif” width=”61” height=”1” border=”0” alt=””></td>
   <td width=”113”><img src=”vista/imagenes/spacer.gif” width=”1” height=”1” border=”0” alt=””></td>
 </tr>
 <tr>
  <td colspan=”2”><img name=”registro_r1_c1” src=”vista/imagenes/registro_r1_c1.jpg” width=”145”
height=”160” border=”0” alt=””></td>
  <td><img name=”registro_r1_c3” src=”vista/imagenes/registro_r1_c3.jpg” width=”159” height=”160”
border=”0” alt=””></td>
  <td colspan=”2”><img name=”registro_r1_c4” src=”vista/imagenes/registro_r1_c4.jpg” width=”146”
height=”160” border=”0” alt=””></td>
  <td><img src=”vista/imagenes/spacer.gif” width=”1” height=”160” border=”0” alt=””></td>
 </tr>
 <tr>
  <td rowspan=”2”><img name=”registro_r2_c1” src=”vista/imagenes/registro_r2_c1.jpg” width=”92”
height=”240” border=”0” alt=””></td>
   <td colspan=”3”>
          <form name=”form1” method=”post” action=”<%=request.getContextPath() %>/registro”>
             <table width=”281” border=”0” cellspacing=”0” cellpadding=”0”>
       <tr>
```

```
          <td width="24" height="33">       
          </td>
          <td width="82"><strong><font size="2" face="verdana">Usuario:</font></strong></td>
          <td width="160"> <input type="text" id="usuario" name="usuario" size="20"> </td>
          <td width="15"><font face="verdana"> </font></td>
        </tr>
        <tr>
          <td height="36"> </td>
          <td><strong><font size="2" face="verdana">Password:</font></strong></td>
          <td><input type="password" id="pass" name="pass" size="20"></td>
          <td><font face="verdana"> </font></td>
        </tr>
        <tr>
          <td> </td>
          <td><font face="verdana"> </font></td>
          <td><input type="submit" name="Submit" value="Enviar"></td>
          <td><font face="verdana"> </font></td>
        </tr>
      </table>
            </form>
            </td>
  <td rowspan="2"><img name="registro_r2_c5" src="vista/imagenes/registro_r2_c5.jpg" width="61"
height="240" border="0" alt=""></td>
  <td><img src="vista/imagenes/spacer.gif" width="1" height="115" border="0" alt=""></td>
 </tr>
 <tr>
  <td colspan="3"><img name="registro_r3_c2" src="vista/imagenes/registro_r3_c2.jpg" width="297"
height="125" border="0" alt=""></td>
  <td><img src="vista/imagenes/spacer.gif" width="1" height="125" border="0" alt=""></td>
 </tr>
</table>
</body>
</html>
```

Tabla 6.2 index.jsp

Como se puede observar en su mayor parte la vista se compone de código HTML, pero es necesario rescatar algunos tramos de código importantes para el funcionamiento de la aplicación.

En la línea donde se declara el form se puede observar que en action de la forma se utiliza un getContextPath, el cual brinda la ruta raíz del sistema web para la liga a la URL del ServeIt, el cual tiene el nombre de *registro* tal y como se puede observar a continuación:

```
<form name="form1" method="post" action="<%=request.getContextPath() %>/registro">
```

Más adelante se observan los objetos de tipo input los cuales son las cajas de texto que sirven de interfaz para que el usuario capture su usuario y password (contraseña). El name de los objetos es el valor que se debe de tener presente debido a que en el controlador se identificará este objeto con esta variable.

```
<input type="text" id="usuario" name="usuario" size="20">
<input type="password" id="pass" name="pass" size="20">
```

A continuación se explicará la parte del modelo, esta sección está formada por dos clases que son Cliente.java y DAOCliente.java.

En la clase Cliente se declaran los atributos, el constructor y los métodos get y set, en la tabla 6.3 se muestran estas declaraciones. La tabla sólo muestra un

método get y uno set pertenecientes al IdCliente, realmente la clase cuenta con una declaración de este tipo por cada atributo pero no se colocarán debido a que todos son iguales.

```
Public class Cliente {
    private int iIdCliente;
    private String sNombre;
    private String sPaterno;
    private String sMaterno;
    private String sTelefono;
    private String sCorreo;
    private int iPerfil;
    private String sUsuario;
    private String sPassword;

    /** Creates a new instance of Cliente */
    public Cliente() {
    }

    public Cliente(int id,String nom,String pat,String mat,String tel,String corr,int per, String usu, String pass)
{
      iIdCliente=id;
      sNombre=nom;
      sPaterno=pat;
      sMaterno=mat;
      sTelefono=tel;
      sCorreo=corr;
      iPerfil=per;
      sUsuario=usu;
      sPassword=pass;
    }

    public void setIdCliente(int id){
        this.iIdCliente=id;
    }
    public int getIdCliente(){
        return this.iIdCliente;
    }
```

Tabla 6.3 Cliente.java

La siguiente parte del modelo es la clase DAOCliente, en esta clase es donde se encuentran los métodos necesarios para que la aplicación interactúe con la base de datos, en la tabla 6.4 se muestra una fracción del código de la clase. Exclusivamente el método registroCliente y el constructor.

```
    /** Creates a new instance of DAOCliente */
    public DAOCliente() {
    }

  public static ArrayList registroCliente(Cliente c){
        ArrayList res = new ArrayList();
        String query;
        Connection con=null;
        con=Conexion.getConexion();
    try{
        Statement stmt = con.createStatement();
        query="SELECT * FROM usuario WHERE usuario='" + c.getsUsuario() + "' and pass='" +
c.getsPassword() + "'";
        ResultSet rs = stmt.executeQuery(query);
        while(rs.next()){
            res.add(new      Cliente(rs.getInt(1),rs.getString(3),rs.getString(4),rs.getString(5),      rs.getString(6),
rs.getString(7),rs.getInt(2),rs.getString(8),rs.getString(9)));
        }
```

```
        stmt.close();
    }catch(SQLException e){
       System.out.println("Error en insertar:" + e.getMessage() );
    }
    Conexion.cerrarConexion(con);
    return res;
    }
}
```

Tabla 6.4 DAOCliente.java

El método que se utilizará para el registro es registroCliente, el cual cuenta con un parámetro de tipo Cliente, es decir, que se genere a partir de la clase Cliente, este método regresa un ArrayList, que contará con toda la información necesaria una vez que algún usuario se haya registrado en el sistema y que la aplicación haya permitido el acceso.

Para esto es necesario crear una conexión a la base de datos, en la tabla 6.5 se muestra el código de la clase Conexion.java.

```
public class Conexion {

   /** Creates a new instance of ConexionBD */
    public Conexion() {
    }

    public static Connection  getConexion(){
    Connection cn = null;
        try{
            Class.forName("com.mysql.jdbc.Driver");
            cn=DriverManager.getConnection("jdbc:mysql://localhost/sgrf","user","user");
        }atch(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;
    }
}
```

Tabla 6.5 Conexion.java

Esta clase sólo se encarga de abrir una conexión contra la base de datos en MySql, como se puede observar existe un método llamado getConexion de tipo Connection, el cual se puede observar en la tabla 6.4 dentro del método registroCliente.

Una vez realizada la conexión a la base de datos se crea una consulta de tipo select donde se especifican el usuario y el password del usuario que intentará el acceso al sistema, esta búsqueda regresará un ArrayList con la información del usuario que se ha registrado, en caso de que los parámetros de entrada hayan sido correctos.

La clase DAOCliente cuenta con más métodos, pero estos se explicarán más adelante, cuando se hable del módulo principal, es decir, la inserción de Fallas dentro del sistema.

Ahora se procederá a explicar la parte del controlador, como se explicó con anterioridad, el controlador es el que interactuará como intermediario entre la vista y el modelo, en la tabla 6.6 se puede ver el código del servelt registro.

```
package com.diploweb.controller;

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

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

import com.diploweb.model.Cliente;
import com.diploweb.model.DAOCliente;
import java.util.*;

public class registro extends HttpServlet {

   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      HttpSession session = request.getSession(true);

      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

      String usuario = request.getParameter("usuario");
      String pass = request.getParameter("pass");

      Cliente oRegistro = new Cliente(0,"","","","","",0,usuario,pass);
      Cliente oCliente = new Cliente();
      ArrayList arrRegistro = DAOCliente.registroCliente(oRegistro);
      if (arrRegistro != null){
         oCliente = (Cliente)arrRegistro.get(0);
         if (oCliente.getIdCliente() != 0){
//             request.setAttribute("oRes", oRegistro);
            session.setAttribute("id", String.valueOf(oCliente.getIdCliente()));
            session.putValue("nombre", oCliente.getsNombre());
            session.putValue("paterno", oCliente.getsPaterno());
            session.putValue("materno", oCliente.getsMaterno());
            session.putValue("correo", oCliente.getsCorreo());
            session.putValue("telefono", oCliente.getsTelefono());
            session.putValue("perfil", String.valueOf(oCliente.getiPerfil()));
            request.getRequestDispatcher("/vista/Principal.jsp").forward(request, response);
         }else{
            request.getRequestDispatcher("index.jsp").forward(request, response);
         }
      }
      else
      {
         request.getRequestDispatcher("index.jsp").forward(request, response);
      }
      out.close();
   }
```

Tabla 6.6 Servlet registro.java

Primeramente, se proporciona el paquete en donde se ubicará el Servlet, todos los controladores se encontrarán en com.diploweb.controller, más adelante se importarán las librerías necesarias que el servlet utilizará para su correcto funcionamiento. Inmediatamente se puede observar que se declara una instancia para sesiones que son sumamente importantes dentro de la aplicación debido a que ellas guardarán el identificador del cliente que se haya registrado.

Después se declaran variables de tipo String que recolectarán los parámetros que se introdujeron en la vista del registro (index.jsp), es decir, los valores introducidos por parte del usuario que se intenta registrar en la aplicación, que son usuario y contraseña (password). Las dos líneas siguientes son importantes, ya que en ellas se crean dos instancias de la clase Cliente utilizando sus constructores; a la primera instancia llamada oRegistro se le enviarán los parámetros necesarios para dar valor a todos los atributos de la clase, en la siguiente línea: Cliente oRegistro = new Cliente(0,"","","","","",0,usuario,pass), se puede observar que todos los parámetros se llenan con cadenas vacías o con ceros esto es debido a que los únicos parámetros que son necesarios son usuario y pass, los cuales son las variables de tipo String que se han recolectado previamente. El siguiente objeto o instancia de clase es oCliente, este se crea utilizando el constructor sin parámetros, y se puede observar en la tabla 6.3.

La primera instancia, oRegistro, se utiliza para enviar el objeto de tipo Cliente hacia el método registroCliente de la clase DAOCliente, esto se guardará dentro de un ArrayList llamado arrRegistro, este contendrá la información de la búsqueda del usuario que se intenta registrar y en caso de que contenga información será asignado al objeto oCliente, aquí es donde se verificará si la información introducida es correcta, y si lo es, el objeto de tipo Cliente tendrá asignado valores a los atributos de la clase, por lo tanto el IdCliente contendrá algún valor, de lo contrario el valor será cero y la información introducida será inválida, direccionando al usuario que se ha intentado registrar nuevamente a la vista, es decir, a la página de registro, index.jsp.

Si el usuario y el password son correctos entonces se generan sesiones para guardar los datos principales del usuario, por ejemplo el IdCliente y el IdPerfil. Por último el sistema enviará al usuario a la página Principal.jsp.

6.2.3 Registrar Falla.

Una vez que el usuario se encuentra en la página principal de la aplicación es necesario seleccionar la liga de registrar falla, para este momento, el sistema ya cuenta con las sesiones necesarias para identificar qué usuario se ha registrado, así como el perfil con el que cuenta.

El módulo de registro de fallas se basa al igual que el registro, en el patrón MVC, por lo tanto también cuenta con un modelo, una vista y un controlador.

Ahora se procederá a explicar la vista del registro de fallas, y solo se mostrará el código importante de la vista es el siguiente:

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="com.diploweb.controller.registro" %>
<%@page import="java.util.*" %>
<%@page import="java.io.*" %>
<%@page import="javax.servlet.*" %>
<%@page import="javax.servlet.http.*" %>
<%@page import="javax.net.*" %>
```

```
<%@page import="java.lang.*" %>
<%@page import="java.text.*" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<%
   int id = (Integer)session.getValue("id");
   String nombre = (String)session.getValue("nombre");
   String paterno = (String)session.getValue("paterno");
   String materno = (String)session.getValue("materno");
   String correo = (String)session.getValue("correo");
   String telefono = (String)session.getValue("telefono");
   String perfil = (String)session.getValue("perfil");

   Date hoy = new Date();
   SimpleDateFormat formateador = new SimpleDateFormat("dd/MM/yyyy");
   String sFecha = formateador.format(hoy);

   if (perfil != 3 | perfil != 4){
      request.getRequestDispatcher("../index.jsp").forward(request, response);
   }
%>
    .
    .
    .
      <form method="POST" action="<%=request.getContextPath() %>/AddFalla" >
         <table  border="0"  width="617"  cellspacing="5"  cellpadding="0"  style="border-collapse:  collapse"
bordercolor="#111111" height="222">
                     <tr class="tituloPrincipal">
               <td colspan="3">Registro de Fallas</td>
                     </tr>
            <tr class="Labels">
               <td width="189"></td>
               <td width="8"> </td>
               <td width="354"></td>
                     </tr>
                     <tr class="Labels">
               <td width="189">
                  Fecha del Reporte:</td>
               <td width="8" > </td>
               <td width="354"><%=sFecha%></td>
            </tr>
            <tr class="Labels">
               <td width="189">Nombre del Cliente:</td>
               <td width="8"> </td>
               <td width="354"><%=nombre + " " + paterno + " " + materno %></td>
            </tr>
            <tr class="Labels">
               <td width="189">Correo Electrónico:</td>
               <td width="8"></td>
               <td width="354"><%=correo%></td>
            </tr>
            <tr class="Labels">
               <td width="189">Teléfono:</td>
               <td width="8"> </td>
               <td width="354" ><%=telefono%></td>
            </tr>
            <tr class="Labels">
               <td width="189"> </td>
               <td width="8"></td>
               <td width="354"> </td>
            </tr>
            <tr class="Labels">
               <td width="189">Título del Reporte de Falla:</td>
               <td width="8"> </td>
               <td width="354"><input type="text" name="txtTitulo" id="txtTitulo" size="51"></td>
            </tr>
            <tr class="Labels">
```

```
            <td width="189">Tipo de Falla:</td>
            <td width="8"> </td>
            <td width="354">
              <select size="1" name="cboTipo" id="cboTipo">
                    <option value="Hardware">Hardware</option>
                    <option value="Software">Software</option>
              </select></td>
          </tr>
    </form>
  .
  .
  .
    </td>
    <td width="23%" height="356" bordercolor="#FFCC00" style="border-left-style: solid; border-left-width: 1; border-right-style: solid; border-right-width: 1" valign="top">
      <%@include file="menu.jsp"%>
    </td>
```

Tabla 6.7 insertarFalla.jsp

Este archivo, primeramente importa algunos paquetes de clases necesarias para su funcionamiento, inmediatamente, se crean variables que reciben el valor de las variables de sesión que se crearon en el módulo de registro. Posteriormente, se genera una variable de fecha, la cual se encarga de mostrar la fecha del sistema y se le da un formato especial; por último se realiza una validación donde se evalúa si el tipo de usuario que se ha registrado tiene permitido el acceso a este módulo de la aplicación, esto es muy importante, debido a que esto es lo que brinda los permisos a los diferentes tipos de perfil del sistema.

Adelante se asigna al action del formulario el servlet AddFalla, al igual que se hizo en el módulo de registro. Por último en lo que respecta a la vista del registro de fallas, se crean los objetos para la recolección de información necesaria para dar de alta una falla: el título y tipo de la falla, sistema operativo, descripción de la falla y prioridad.

Una línea que es necesario recalcar es la siguiente: <%@include file="menu.jsp"%>; su importancia radica en que con esto se incluye el menú de la aplicación el cual es un archivo de tipo jsp donde se encuentran todas las URL's (ligas) para la navegación dentro del sistema. Cabe destacar que todas las páginas (vistas) de la aplicación contarán con esta inclusión para mostrar el menú.

Ahora se explicará el modelo del módulo de alta de fallas. El modelo se divide en dos partes, cada una de estas partes es una clase, la primera es la clase ReporteT.java y la otra es Falla.java, a su vez cada una de estas partes contiene su clase para la convivencia con la base de datos (de igual forma que en el módulo de registro), y estas son DAOReporteT.java y DAOFalla.java.

La clase RegistroT es una superclase de donde heredarán las clases Falla y Requerimiento. A continuación se explicará el código de la clase ReporteT y posteriormente se procederá a hacer lo mismo con Falla.

En la tablas siguientes (6.8 y 6.9) se muestra el código de las clases ReporteT y Falla respectivamente.

```
public class ReporteT {
    private int idReporte;
    private int idUsCap;
    private int idUsRes;
    private String fecCaptura;
    private String fecRespuesta;
    private String titulo;
    private String desc;
    private int prioridad;
    /** Creates a new instance of ReporteT */
    public ReporteT() {
    }
    public ReporteT(int idreporte, int iduscap, int idusres, String feccaptura,
            String fecrespuesta, String titulo, String desc, int prioridad){
        this.idReporte = idreporte;
        this.idUsCap = iduscap;
        this.idUsRes = idusres;
        this.fecCaptura = feccaptura;
        this.fecRespuesta = fecrespuesta;
        this.titulo = titulo;
        this.desc = desc;
        this.prioridad = prioridad;
    }


    public void setIdRegistro(int idreporte){
        this.idReporte = idreporte;
    }
    public int getIdReporte(){
        return this.idReporte;
    }
}
```

Tabla 6.8 ReporteT.java

```
public class Falla extends ReporteT{
    private int idFalla;
    private String sOperativo;
    private String tipoFalla;
    private String solucion;

    /** Creates a new instance of Falla */
    public Falla(){
    }
    public Falla(int idreporte, int iduscap, int idusres, String feccaptura, String fecrespuesta, String titulo,
String desc,
            int prioridad, int idfalla, String soperativo, String tipofalla, String solucion){
        super(idreporte, iduscap, idusres, feccaptura, fecrespuesta, titulo, desc, prioridad);
        this.idFalla = idfalla;
        this.sOperativo = soperativo;
        this.tipoFalla = tipofalla;
        this.solucion = solucion;
    }

    public void setIdFalla(int idFalla){
        this.idFalla = idFalla;
    }
    public int getIdFalla(){
        return this.idFalla;
    }
}
```

Tabla 6.9 Falla.java

Al igual que la clase Cliente (Tabla 6.3), Falla y ReporteT sólo son una especie de esqueleto, que tiene los atributos, los constructores y los métodos get y set

de la clase, la tabla 6.8 sólo muestra un método get y un set para ejemplificar la acción.

Una vez mostrados los esqueletos, se procede a explicar las clases DAO, que, al igual que para Cliente, se encargan de realizar las operaciones necesarias de interacción con la base de datos de MySql.

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

public class DAOReporteT {

   /** Creates a new instance of DAOReporteT */
   public DAOReporteT() {
   }
   public static int maxReporte(){
      String query;
      Connection con=null;
      con=Conexion.getConexion();
      int iMaximo=0;
   try{
      Statement stmt = con.createStatement();

      query = "SELECT max(idReporte) FROM reportet;";
      ResultSet rs = stmt.executeQuery(query);
      while(rs.next()){
         iMaximo = rs.getInt(1);
      }
      stmt.close();

   }catch(SQLException e){
     System.out.println("Error en el maximo registro:" + e.getMessage() );
     return 0;
   }
   Conexion.cerrarConexion(con);
   return iMaximo;
   }
}
```

Tabla 6.10 DAOReporteT.java

DAOReporteT, tabla 6.10 cuenta con un único método el cual tiene la función de adquirir el identificador máximo para la tabla de reportet dentro de la base de datos. Al igual que la clase DAOCliente se apoya de la clase Conexion, para realizar la interacción con MySql.

Por otro, lado DAOFalla cuenta con varios métodos, pero para este módulo solo nos enfocaremos en el método insertaFalla, que es el que se encarga de guardar los registros nuevos capturados por algún cliente.

Este método realiza tres operaciones, la primera es realizar una inserción de registro en la tabla reportet; esta tabla guarda los datos compartidos en fallas y requerimientos. Una vez que existe un registro nuevo en reportet, es necesario guardar un registro particular en la tabla falla, el cual complementará la información exclusiva de un reporte de falla capturado por un cliente.

Para que las tablas reportet y falla tengan relación entre sus registros es necesario una llave, un identificador único, el cual se genera dentro de la tabla reportet, pero debido a que este número se genera automáticamente desde la base de MySql es necesario buscar primero el último registro que se haya insertado, ya que este será el que se le asignará como llave foranea a la tabla de falla, para esto se requiere del método maxReporte, que se encuentra en la clase DAOReporteT, el cual se puede consultar en la tabla 6.10.

La tabla siguiente muestra el una fracción del código de la clase DAOFalla solo el método insertaFalla.

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

public class DAOFalla {
      public static Falla insertaFalla(Falla f){
       String query;
       Connection con=null;
       con=Conexión.getConexion();
       int iMaximo=0;
       if (con==null){
          return null;
       }
   try{
       Statement stmt = con.createStatement();
       Statement stmtF = con.createStatement();
       query="INSERT INTO reportet (idUsuarioCap, FecCaptura, Titulo, Descripcion, Prioridad) values ("+
             f.getIdUsCap() +"','"+ f.getFecCaptura() +"','"+f.getTitulo()+"','"+
             f.getDesc()+"','"+f.getPrioridad()+"');" ;
       stmt.executeUpdate(query);
       stmt.close();
       iMaximo = DAOReporteT.maxReporte();
       atch = "INSERT INTO falla (idReporte, Soperativo, TipoFalla, Solucion) values (" + iMaximo +
             ",'" + f.getOperativo() + "','" + f.getTipoFalla() + "','" + f.getSolucion() + "');";
       stmtF.executeUpdate(query);
       stmtF.close();
       f.setIdRegistro(iMaximo);
   }atch(SQLException e){
      System.out.println("Error en atch er:" + e.getMessage() );
      return null;
   }
   Conexión.cerrarConexion(con);
   return f;
   }
}
```

Tabla 6.11 DAOFalla.java

Una vez explicadas las partes de la vista y el modelo, sólo queda el controlador, este se encuentra en la clase AddFalla.java (tabla 6.12); al igual que en el registro, el controlador es un servelt que se encarga de crear las instancias necesarias, de los métodos que existen en la parte del modelo.

Como primer paso, se recuperan los parámetros enviados por el formulario del jsp insertaFalla, estos parámetros cuentan con la información que el cliente capturó para levantar un reporte de falla. Posteriormente se crea una variable de tipo string con la fecha y se le da el formato necesario para ser guardada; a

continuación se crea la instancia para dos objetos de tipo falla, el primero “cF” con el constructor que no posee parámetros y el segundo “oFalla” que si posee. (ver tabla 6.9) El objeto oFalla, recibe todos los parámetros enviados por el formulario del jsp y después se utiliza para ser enviado como parámetro al método insertaFalla de la clase DAOFalla, este método como ya se explicó, se encarga de realizar la inserción de registros en la base y regresa un objeto de tipo Falla que es asignado a la instancia cF, se evalúa si cF es diferente de nulo para comprobar que el método haya realizado todo de forma correcta, de ser así, se enviará al usuario de aplicación al jsp RegistroResultado enviando un parámetro de tipo Falla llamado “oFalla” que contiene la información contenida en cF; en caso de que cF sea nulo la aplicación envía al usuario a una página de error

Es necesario comentar que en esta parte es cuando entraría en acción la clase Mail, esta clase sería la encargada del envío de un correo electrónico a los técnicos correspondientes, donde se informaría que existe un nuevo reporte de falla, este código no se muestra debido a que el alcance del desarrollo no ha permitido su realización, pero ya que en el diagrama de casos de uso se menciona esta actividad, se explica brevemente como sería su funcionamiento.

```
package com.diploweb.controller;

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

import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.text.*;
import com.diploweb.model.DAOFalla;
import com.diploweb.model.Falla;

public class AddFalla extends HttpServlet {
   /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      HttpSession session = request.getSession(true);

      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

      //Sesiones
      String id = (String)session.getValue("id");
      int idSesion = Integer.parseInt(id);

      //Valores obtenidos de objetos del formulario
      String titulo=request.getParameter("txtTitulo");
      String tipo=request.getParameter("cboTipo");
      String sistema=request.getParameter("cboSistema");
      String desc=request.getParameter("txaDesc");
      int prioridad=Integer.parseInt(request.getParameter("cboPrioridad"));

      Date hoy = new Date();
      SimpleDateFormat formateador = new SimpleDateFormat("yyyyMMdd");
      String sFecha = formateador.format(hoy);

      Falla cF = new Falla();
```

```
        Falla oFalla = new Falla(0,idSesion,0,sFecha,"",titulo,desc,prioridad,0,sistema,tipo,"");

        cF = DAOFalla.insertaFalla(oFalla);
        if(cF != null){
            request.setAttribute("oFalla", cF);
            request.getRequestDispatcher("/vista/RegistroResultado.jsp").forward(request, response);

        }else{
            request.getRequestDispatcher("/vista/Error.jsp").forward(request, response);

        }

        out.close();
    }
```

Tabla 6.12 Servlet AddFalla.java

En este momento el cliente o usuario que ha decidido dar de alta una falla se encuentra en la página RegistroResultado.jsp con un aviso de que su información ha sido insertada correctamente y algunos datos del reporte que levantó, así como un identificador de número de reporte para cuando este sea contestado, sea posible su búsqueda.

Debido a que este es sólo un prototipo, no se desarrollaron todos los módulos del sistema y sólo fue desarrollado el módulo de registro de fallas, el cual debe contar también con una clase más llamada Mail, que se encargaría de dar aviso a algún técnico de que existe un nuevo reporte de falla. El correo electrónico recibido por el técnico debería contener los mismos datos que fueron dados al cliente que registró la falla, mismos que fueron impresos en la página RegistroResultado.jsp. Por supuesto, el dato de mayor relevancia es el número identificador del reporte.
Una vez desarrollado el primer módulo del sistema, los demás tendrían una estructura muy similar y su realización sería rápida y sin dificultad.

Por ejemplo, para el desarrollo de la parte de usuarios, se estarían utilizando los métodos pertenecientes a la clase Cliente, la cual como se ha visto se divide en dos que son Cliente y DAOCliente. Los objetos se crearían con los constructores de Cliente y sería necesaria la creación de los servlets para la interacción de las interfaces con las clases del modelado.

Conclusiones.

La Aplicación.

Hoy en día, la tecnología avanza a pasos agigantados y con la llegada de Internet a nuestras vidas, algunas metodologías de trabajo han quedado un tanto obsoletas.

Un ejemplo de esto son los diarios o periódicos; ya existen diarios electrónicos como El Universal en línea (www.eluniversal.com.mx) que aunque no ha logrado desplazar por completo a la versión impresa, es un hecho que la supera por completo debido a que la información es actualizada al momento y además aporta un beneficio a la ecología por el ahorro de papel, lo que reduce significativamente la tala de árboles. Otro ejemplo claro de esta evolución son los correos electrónicos que prácticamente han desplazado al envío de cartas tradicional o los servicios de mensajería instantánea, ya que hacen posible la comunicación entre una persona aquí en México y otra que se encuentre al otro lado del mundo en tiempo real, y no es posible sólo el envío de mensajes de texto, también es posible la comunicación vía voz y video.

Debido a esta creciente demanda de información se propuso un sistema que intenta tomar toda esta nueva forma de ver el mundo y optimizar el flujo de información entre un proveedor y sus clientes.

Es muy sabida la existencia de empresas que se dedican a la fabricación de software, algunos ejemplos son Microsoft y Oracle. Estas empresas debido a su gran infraestructura cuentan con metodologías y desarrollos tecnológicos para brindar a sus clientes un gran servicio; pero también existen muchas empresas de tamaño mediano y pequeño (consultorías) que no cuentan ni con la infraestructura, ni con los desarrollos y ni siquiera con las metodologías apropiadas para el manejo de información entre ellos y sus clientes.

Aquí es donde entra el sistema para la gestión de fallas y requerimientos; con la intención de apoyar a estas empresas para que cuenten con una mejor calidad de servicio y atención.

Como se observó el sistema está basado en tecnología Java para WEB (Internet). Por lo que la aplicación en línea permite la rápida transferencia de quejas (fallas) o solicitudes (requerimientos) de un cliente hacia su proveedor, así como las respuestas por parte del proveedor. Guardando, además, un registro histórico de reportes que permiten la evaluación de nivel de atención a los clientes, así como su satisfacción.

Por último, cabe destacar que la aplicación con algunas pequeñas modificaciones podría acoplarse a otros tipos de empresas; por ejemplo, una empresa de electrodomésticos, en donde algún cliente pueda hacer valer la garantía de un aparato eléctrico y concertar una cita para la visita de un técnico a su hogar. Otro ejemplo, lo pedemos observar en un restaurante donde los clientes podrían solicitar en línea algún pedido.

El Desarrollo y la Implementación.

El trabajo como se pudo observar en los capítulos dos y tres, habla sobre la ingeniería del software y los procesos para el desarrollo del mismo. Para esta aplicación se implementó la metodología de XP (extreme programming) la cual optimiza tiempos en la elaboración de aplicaciones. Se implementó debido a que la intención principal era conseguir los objetivos lo antes posible y con el menor tiempo invertido en requerimientos y planeación junto con el cliente. Esta metodología resultó de gran utilidad, debido a que el proyecto a realizar así lo permite, y es posible ir recopilando requerimientos sobre la marcha. Por otro lado, se recomienda una metodología mucho más robusta y elaborada; como RUP, PSP o TSP (ver capitulo 3); cuando se intente fabricar aplicaciones grandes y con procesos demasiado complicados o donde deben interactuar gran cantidad de desarrolladores.

Debido a que se trata de una aplicación muy sencilla el trabajo puede orientar a programadores principiantes a dar sus primeros pasos para introducirse en el ambiente del desarrollo de sistemas.

Definitivamente la mejor forma de acelerar el aprendizaje y el desarrollo de un sistema como éste es la práctica, una vez poseídos los conceptos generales del desarrollo de un sistema web es necesario practicar y practicar para lograr afinar de mejor forma los códigos de la aplicación, así como la metodología de desarrollo.

También es necesario desarrollar el sitio web con el control total, es decir, ser capaz de probar soluciones, confirmarlas y aplicarlas en ciclos breves que favorezcan ganar el "momento" y mostrar respuestas a las demandas de los usuarios.

Definitivamente lo más importante es tener pleno conocimiento de los requerimientos para el desarrollo del sistema, una vez dominadas y digeridas las reglas del negocio, el desarrollo sistemático y analítico será mucho más sencillo.

Por último es muy importante tomar en cuenta que es imposible tomar decisiones si no se ha participado en el desarrollo de forma directa y no se han evaluado las consecuencias de los actos.

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 en 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