UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

Arquitectura de sistemas empresariales J2EE implementando software libre en la creación de aplicaciones de misión crítica.

TESIS

QUE PARA OBTENER EL TITULO DE

Licenciado en Matemáticas Aplicadas y Computación

PRESENTA

Percival De León Salgado

Asesor: Javier Rosas Hernández

Marzo 2007

Agradecimientos:

Primeramente a Dios por derramar infinidad de bendiciones sobre mí…

A mis padres Víctor Joaquín y Maria Angelina por darme todo su apoyo y la base de mi crecimiento…

A mi linda Andrea por ser la chispa adecuada en mi vida…

A Javier Baltierrez Castillo por ser el mejor mentor, amigo y colega…

A mi asesor Javier Rosas Hernández por su paciencia, sugerencias y confianza…

A mis familiares y amigos por su apoyo incondicional…

ÍNDICE

Introducción

En el ámbito computacional la evolución es cada vez mayor, no sólo en el aspecto de las nuevas tecnologías de información, sino también en la filosofía de ellas. Muchas de estas filosofías apenas están naciendo y están revolucionando el pensamiento de sus usuarios. Una de ellas es la compra de software con muchas restricciones antipiratería, que hace al software un "producto" hermético, el cual no permite modificaciones de su funcionamiento ni adición de nuevas funcionalidades sin el debido consentimiento del creador del software que en ocasiones es imposible obtener un permiso del estilo. No sólo eso, si no que su costo es muy elevado y en ocasiones el costo depende de la capacidad de usuarios que lo vayan a utilizar. La filosofía de pago suele ser recursiva, es decir, después de determinados periodos se tiene que estar pagando, en ocasiones de por vida.

Conforme pasa el tiempo, cada vez se hacen más presentes métodos para automatizar, diseñar o crear algo, esto tiene como finalidad el resolver problemas. Como consecuencia, muchas personas, compañías y asociaciones civiles opten por organizar a nivel mundial un movimiento el cual tenga como fin desarrollar lenguajes, aplicaciones, sistemas manejadores de base de datos, etc., con absoluta libertad en cuanto a costo, disponibilidad del código, funcionalidad y portabilidad.

A lo largo de mi vida como estudiante e investigador me he dado cuenta de lo brillante que llega a ser el ser humano cuando se propone a renovar, inventar o modificar algo. Cuando su Fe por el funcionamiento de determinada cosa no cesa hasta verla funcionar y no sólo eso si no que de la mejor manera, para que ésta no falle. Toda esta idea surgió de la inquietud por analizar, diseñar, construir software de clase mundial y aplicar todo el conocimiento adquirido en mi formación como profesionista en la elaboración de software adoptando una filosofía de uso, que está basada en la del software libre.

También deseo difundir la admirable labor de entidades como la Free Software Foundation (FSF) y la de cientos de voluntarios unidos a través de Internet, que desarrollan y comparten el software libre, que día a día atienden necesidades cada vez más grandes y más complejas.

El Software Libre se hace más popular a pesar de que no ha tenido la publicidad que tiene el software propietario principalmente porque el principal objetivo no es comercial. El Internet es la principal herramienta de comunicación y difusión para este medio. Es en el Internet en donde se observa el gran auge que esta teniendo el software libre debido a que la mayor parte de software que se mueve en Internet es Software Libre.

Los sistemas basados en Web son una opción para resolver necesidades de implementación de una aplicación que automatice cálculos y procedimientos todos ellos utilizados con protocolos de red, donde la utilización de estos se basa en la funcionalidad que dan las redes computacionales.

En esta investigación el objetivo es proponer una arquitectura de solución la cual pueda ser la base de crecimiento tecnológico de una empresa, permitiendo automatizar procesos sin la

necesidad de representar un gasto en compra y mantenimiento de licencias de software para resolver este tema.

Los capítulos presentados son:

- Acerca de Software Libre
- Arquitectura de aplicaciones empresariales en capas basado en el estándar J2EE
- Propuesta de una arquitectura de sitemas empresarial en capas totalmente basada en Software Libre
- Aplicación de la arquitectura J2EE utilizando software libre en aplicaciones de misión crítica

En el capítulo 1 se analizará el tema de software libre desde la perspectiva de sus fundadores (como Richard Stallman) asi como las diferentes definiciones de lo que envuelve al tema del software libre, el incremento en el uso de software basado en esta filosofia y sus características más importantes.

En el capítulo 2 se explorarán las características y capacidades de las aplicaciones basadas en el estandar J2EE, así como también de los factores más importantes relacionados con la Web.

En el capítulo 3 se propondrá una arquitectura de n-capas utilizando el estadar J2EE basada en la filosofia del software libre, capaz de servir soluciones de nivel empresarial manejando información crítica de ella.

En el capitulo 4 se explicará el funcionamiento de un software libre llamado Compiere trabajando con una arquitectura totalmente basada en software libre. La propuesta de solución a tocar en el capítulo 3 tendrá la capacidad de funcionar de manera óptima con Compiere. El objetivo de presentar la funcionalidad de Compiere es el ejemplificar los alcances que puede tener un software libre en una arquitectura robusta y pensada para ser un ejemplar en el manejo de procesos críticos de una empresa.

El objetivo de esta investigación se enfoca en las aplicaciones de misión crítica, las cuales debido a la importancia que tienen al mantener un servicio operable en una empresa, la aplicación debe tener el diseño y software capaz de mantener la disponibilidad del servicio.

Como diría la popular frase:

"La necesidad es la madre de todos los inventos"

Capítulo 1

1 Acerca del Software Libre

1.1 Antecedentes del Software Libre

La sociedad de los países industrializados tiene una serie de características que la hacen destacar sobre otras sociedades: una renta per cápita superior a la media mundial, un desplazamiento de su actividad productiva hacia el sector servicios, una serie de valores económicos y culturales "occidental". Una vez caído el sistema productivo "comunista", la sociedad de mercado con sus ventajas e inconvenientes se ha hecho con la supremacía de las sociedades industrializadas, y con esto un fenómeno que en los últimos años se ha hecho patente: vivimos en la sociedad de la información. Tanto es así que nuestro modelo económico se basa en muchas ocasiones para su desarrollo, en la posesión y manejo de información valiosa o privilegiada. Una frase de periodistas es "Quien tiene la información, tiene el poder", pero este tipo de concepto tiene otra realidad "El poder no está en quien tiene la información, sino en quien sabe manejarla. "

En un mundo con superabundancia de fuentes de información es un problema real el manejo y proceso de tanta información. Tal es así que podríamos decir que hasta estamos viviendo una era de sobre información y como ejemplo tenemos al medio más grande o si no uno de los mas grandes: "Internet."

Siguiendo este contexto de información podemos decir que el Internet viéndolo desde una perspectiva sociológica, ha convertido a este fenómeno de información y el acceso a ésta, en un método o mecanismo en el que todos pueden tener acceso con un mínimo de recursos, pareciera como si todo esto se hubiera convertido en una especie de "democratización" sin tener en cuenta el valor de la información, dentro del Internet podemos clasificar a la información en:

- Publicidad
- Juegos ,Viajes, Turismo, Entretenimiento con una variedad de tipos, etc
- Cultural
- Socio-Laboral
- Información Científica y Técnica
- Materiales y Técnicas de cómo buscar más información(Meta –Información)
- etc.

Entonces, el usuario tiene de dos opciones ante todo este concepto de información, una es intentar buscarla y otra generarla por si mismo, que sin importar cual de los dos caminos se decida tomar, se genera un coste que para una empresa se traduciría como un coste económico. Suponiendo que se toma el segundo camino, la empresa tendría que invertir tiempo y dinero para la elaboración de la información y en el caso del primero la empresa invertiría tiempo en que sus

empleados encuentren y procesen la información para que ésta sea útil. Pero tomando cualquiera de los dos caminos implica que la información se filtre, procese y asimile varias veces.

Una vez que la empresa encuentra la información que necesita, se plantea la necesidad de darle una salida. Para eso tenemos dos modalidades una es la "CERRADA" (el resultado de su trabajo es de uso exclusivo de la empresa) y la otra "ABIERTA" (se comparte -de forma aprovechable- el resultado del trabajo de la empresa). Entonces, podemos decir que una empresa cuando toma la modalidad de Abierta lo denomina un modelo empresarial tipo "Software Libre"

El software libre es tan antiguo (se podría decir que más que Internet). De hecho podemos decir que Internet no existiría sin el software libre. Desde que en los años 60 Los Bell Telephone Laboratorios de AT&T cedieron el código fuente de su recién inventado Sistema Operativo UNIX (que en esos años adquirió su primer nombre como "Multics" para después pasar por una serie de cambios hasta convertirse en UNIX).

Una serie de características hacen en la industria del software la necesidad de una comunicación fluida y de un intercambio de información:

- La necesidad de formatos de intercambio de datos entre distintas aplicaciones
- Los protocolos de comunicaciones entre sistemas, que exigen por un lado la existencia de un API, o modelo de programación estándar, y por otro lado una "implementación tipo" o modelo de referencia a quien seguir
- Los formatos propietarios son rechazados casi sistemáticamente en los entornos comerciales, debido a que ningún comerciante que se precie, y que tenga un mínimo deseo de subsistir, va a adoptar formatos que no sean manejables por todos sus clientes potenciales.

No obstante, intereses comerciales hacen que muchos fabricantes quieran imponer sus propios estándares, o añadan extensiones incompatibles a los ya existentes.

- El mundo del software abierto es capaz en poco tiempo de adaptar el código libre para soportar extensiones no estándar a los protocolos. Todo ello se traduce para la empresa que desarrolla dichos protocolos en una carrera por inventar nuevas extensiones, hasta que se llega a la inutilidad de los esfuerzos realizados en forzar un estándar.

- El desarrollo de extensiones propietarias tiene además un riesgo tecnológico: la carencia de un modelo de referencia da lugar a la posibilidad de cometer errores, difíciles de detectar y corregir -dada la naturaleza "propietaria" del código-, así como la desconfianza del usuario ante "puertas ocultas" o fallos de seguridad. Los errores de codificación de implementaciones propietarias han sido la constante en muchos paquetes de software, por ejemplo el sistema operativo más comercial del mundo "Windows" y la mayoría de los productos de la empresa Microsoft.

- El empresario que posee un mercado cautivo sobre los usuarios de dicho software muchas veces tiende a ignorar a los usuarios. Del mismo modo, muchos usuarios admiten los errores como algo consustancial al mundo del software. Esta realimentación negativa, a la larga redunda en una pérdida de eficiencia y de productividad del usuario, y a una falta de motivación para la mejora de la calidad en la empresa de software. Finalmente, se acaba produciendo una "desconexión" entre el cliente y su proveedor: el producto deja de responder a las expectativas del usuario, éste deja de "realimentar" a su proveedor y se acaba adoptando otro software similar que realmente responde a las expectativas. Mientras tanto, el proveedor y el usuario sufren pérdidas económicas por el uso de un producto y una metodología que no son capaces de satisfacer a todos los elementos que intervienen en la cadena

- El mercado del software es un auténtico campo de batalla: podemos hablar de "ecosistemas", donde sólo cabe un competidor. No existe tiempo para desarrollar más que nuevas prestaciones para un producto. Es mucho más efectivo, duradero y rentable, capturar clientes con un buen producto que con un API propietario. Además, esta última táctica dirige masa productiva a actividades no directamente remuneradoras, sino "de protección", que a la larga, se revelan ineficaces. Sólo es efectiva esta estrategia en líneas de desarrollo a muy corto plazo, en las que no tiene sentido para la competencia desarrollar contramedidas

- Recientes hechos y actuaciones judiciales ponen en punto de controversia el software propietario, en el sentido de que en algunos casos se deriva en situaciones de falta de defensa para el usuario final: bases de datos de usuarios, archivos ocultos, información confidencial acaba llegando al creador del producto, sin que el usuario pueda hacer nada para evitarlo

Lo que se indica con todo esto es que sólo cuando es asumible el coste de la innovación plantearse la creación de protocolos o interfaces propietarios, el mercado tiene gran inercia y necesita apoyarse en bases conocidas, por lo que ninguna empresa media permite el lujo de forzar abrir mercado (clientela) dentro de un mercado competitivo. La adopción de estándares conocidos (y aceptados) permite a los usuarios arriesgarse a probar el nuevo producto, y la existencia de modelos de implementación públicos permite a la empresa un desarrollo sin necesidad de partir "de cero". Incluso con gigantes de la industria de software se ha dado lugar a problemas muy serios -incluso judiciales- por algo tan simple como la compatibilidad hacia atrás entre dos versiones del mismo paquete informático.

Después de todo lo explicado llegamos a una punto donde se hace visible la necesidad de partir de un estándar aceptado por todos (facilidad para el usuario) y de un modelo de implementación (facilidad para la empresa). Esta característica es esencial en el software libre: Un sistema de producción que favorece a todas las partes implicadas.

No podemos olvidar que el auge de la informática y de las tecnologías de la información es relativamente reciente: hace tres décadas el mundo del software estaba reducido al ámbito académico -y a veces militar- de unos pocos países privilegiados. El proyecto Arpanet fue en su origen un desarrollo de la universidad de Berkeley encargado por el Departamento de Defensa. Las leyes federales obligaron a que después de un tiempo, todo proyecto financiado con fondos públicos, sea del dominio público.

En otros casos no fue la ley, sino los convenios entre empresas y organismos públicos los que potenciaron la distribución de software libre: Si ATT no hubiera cedido el código fuente de su sistema operativo UNIX, posiblemente Internet (si existiera) sería totalmente distinta de lo que conocemos hoy en día.

El sistema X-Windows es mantenido y desarrollado por un consorcio de empresas que se comprometen a seguir estándares y normas de desarrollo comunes, obteniendo a cambio una garantía de operabilidad interna y de acceso a recursos comunes, especialmente el código fuente. Se ha producido recientemente un intento de "privatizar" este club, estableciendo una "cuota de socio", intento que por motivos nuevamente económicos ha sido abortado.

Los intercambios de información han traído consigo preocupaciones de carácter mercantil: la protección de los derechos de autor. Estaba claro que un sistema de producción basado en compartir información debía establecer una serie de garantías sobre los derechos de uso de dicho software. Las primeras pruebas en este sentido se limitaban a garantizar que todo trabajo basado en código abierto debía llevar consigo los créditos del copyright del autor. No se establecían protocolos de protección de la integridad del trabajo o de garantizar la publicidad de todo trabajo derivado. Igualmente, tampoco se imponían restricciones de índole comercial, o de distribución.

De hecho históricamente se ha podido comprobar que la ausencia de reglamentación sobre la propiedad del código abierto ha redundado en un "cierre progresivo" de dicho software, creación de extensiones particulares o incluso la negación de derechos de copyright cuando el código original era relegado a ser una mínima parte del nuevo desarrollo. Nadie puede olvidar que el API de Microsoft Windows fue un convenio de colaboración con Apple para el desarrollo del porting de una aplicación informática al entonces naciente entorno PC. El actual declive de los sistemas UNIX (salvo la excepción de Solaris) no es sino el resultado de un exceso de actitudes de "abuso del cliente" tanto en relación precio/prestaciones como en falta de operatividad interna. Incluso hoy en día, los intentos de unificación (UNIX System V, Common Desktop Environment, etc.) están condenados al fracaso debido a la negativa de las partes a una colaboración efectiva.

La primera generación de software abierto murió de egoísmo y falta de organización (aspecto fundamental indispensable para querer realizar cualquier cosa satisfactoriamente), dando cabida a los sistemas propietarios, que si bien eran productos de desarrollo cerrados, constituían un refugio para el usuario en cuanto a garantía de estabilidad y continuidad. Esta situación pronto degeneró en una situación de monopolio, y de abuso sobre el cliente

Ante todo esto tenía que llegar la reacción: ante un producto que el usuario necesita para su productividad y cuando dicho producto tiene que ser adquirido en condiciones abusivas para el

vendedor, la reacción es obvia: hoy en día la piratería informática mueve más del 60% del mercado mundial de software.

La piratería en nuestros días (ver tabla T1.1) se muestra las cantidades Software pirata como porcentaje del software total instalado en el país, según estimaciones de BSA (Business Software Alliance):

SUR AMERICA		
Argentina (ar)	77,00%	fuente - BSA
Bolivia (bo)	83,00%	fuente - BSA
Brasil (br)	64,00%	fuente - BSA
Chile (cl)	66,00%	fuente - BSA
Colombia (co)	57,00%	fuente - BSA
Paraguay (py)	76,00%	fuente - BSA
Perú (pe)	61,00%	fuente - BSA
Uruguay (uy)	66,00%	fuente - BSA
Venezuela (ve)	82,00%	fuente - BSA
CENTRO AMERICA		
Costa Rica (cr)	66,00%	fuente - BSA
El Salvador (sv)	81,00%	fuente - BSA
Guatemala (gt)	81,00%	fuente - BSA
Honduras (hn)	75,00%	fuente - BSA
Nicaragua (ni)	80,00%	fuente - BSA
NORTE AMERICA		
México (mx)	65,00%	fuente - BSA

Tabla T1.1 Software instalado de origen pirata (fuente BSA, seccion México)

Según BSA pierde México millones de Dólares por piratería de acuerdo al último estudio global

- La piratería en México continúa siendo alta y se ubica sin variación en un 65%
- América Latina la segunda región en el mundo con más piratería de software
- Pérdidas mundiales exceden los $34 000 millones de dólares

Las pérdidas por piratería en Brasil y México –los dos mayores mercados de TI en la región– crecieron más de 100 millones de dólares cada uno, mientras que la tasa de piratería en ambos

países se mantuvo estable, en 64% y 65% respectivamente. Otros países latinoamericanos donde la tasa de piratería se mantuvo igual al año anterior son: Honduras (75%), Nicaragua (80%), Paraguay (83%), Perú (73%) y República Dominicana (77%).

Las pérdidas mundiales por piratería de software llegaron a 34.000 millones de dólares en el 2005, un incremento de 1.600 millones sobre el año anterior. En países con grandes mercados de software, comparativamente las tasas de piratería bajas pueden llegar a grandes pérdidas. Mientras que los Estados Unidos tenían la tasa de piratería más baja de todos los países estudiados, 21 por ciento, también registró las mayores pérdidas individuales -6.900 millones de dólares. China registró el segundo lugar en volumen de pérdidas, con 3.900 millones, con una tasa de piratería de 86 por ciento, seguida por Francia con pérdidas de 3.200 millones y una tasa de piratería de 47 por ciento.

La Business Software Alliance, está conformada por los más grandes productores de software a nivel mundial y sus miembros activos en México son: Adobe Systems Incorporated, Autodesk, Inc., Macromedia Inc., Bentley Systems, Microsoft Corporation y Symantec Corporation.

Como Richard Stallman (Pionero del movimiento de Software Libre) ha mencionado en repetidas ocasiones que: "Cuando el modelo de producción obliga a que más de la mitad de los consumidores actúe en la ilegalidad, es que algo falla en el sistema". De esta idea surge la segunda oleada del software libre, que tiene sus orígenes en la Free Software Foundation.

- La primera premisa es la garantía de la propiedad: Al contrario de lo que muchos pregonan, este modelo de software libre no renuncia a la propiedad de la información, sino que la garantiza hasta extremos que alguno considera abusivos. No solo el trabajo original es propiedad del autor, sino que la licencia de distribución obliga a que todo trabajo derivado del original retenga el copyright de éste.

- La segunda premisa es más sutil: garantiza la continuidad del carácter abierto de la información. Esto que a primera vista parece loable esconde dos motivos ocultos y llenos de repercusiones económicas. La primera es una garantía de que nadie podrá apropiarse del trabajo con fines particulares. La segunda, garantiza la distribución del trabajo hasta unos extremos que harían el paraíso de cualquier director de mercadotecnia de una empresa comercial

En torno a este modelo de desarrollo y a esta filosofía ha nacido lo que hoy constituyen la estrella de los sistemas abiertos: por un lado el núcleo del sistema operativo Linux, y por otro, el conjunto de utilidades GNU de la Free Software Foundation.

1.2 Organizaciones y Proyectos Pro-Software libre

1.2.1 Richard Stallman

Richard Matthew Stallman (a quien se hace referencia comúnmente por sus iniciales RMS) es una figura central del movimiento de Software Libre. Sus mayores logros como programador incluyen

el editor de texto Emacs, el compilador GCC, y el depurador GDB, bajo la rúbrica del Proyecto GNU. Pero su influencia es mayor por el establecimiento de un marco de referencia moral, político y legal para el movimiento de Software Libre, como una alternativa al desarrollo y distribución de software propietario. Stallman nació en 1953 en Manhattan, EUA.

En 1971, siendo estudiante de primer año en la Universidad de Harvard, Stallman se convirtió en un hacker del Laboratorio de inteligencia artificial del MIT. Durante los ochentas, la cultura hacker que constituía la vida de Stallman empezó a disolverse bajo la presión de la comercialización en la industria del software. En particular, otros hackers del Laboratorio de IA fundaron la compañía Symbolics, la cual intentaba activamente reemplazar el Software Libre del Laboratorio con su propio software propietario. Por dos años, desde 1983 a 1985, Stallman por sí solo duplicó los esfuerzos de los programadores de Symbolics para prevenir que adquirieran un monopolio sobre los computadores del Laboratorio. Por ese entonces, sin embargo, él era el último de su generación de hackers en el Laboratorio.

Se le pidió que firmara un acuerdo de no revelado (non-disclosure agreement) y llevara a cabo otras acciones que él consideró traiciones a sus principios. En 1986, Stallman publicó el Manifiesto GNU, en el cual declaraba sus intenciones y motivaciones para crear una alternativa libre al sistema operativo Unix, el cual nombró GNU (GNU no es Unix). Poco tiempo después se incorporó a la organización no lucrativa Free Software Foundation(FSF) para coordinar el esfuerzo. Inventó el concepto de copyleft el cual fue utilizado en la Licencia Pública General GNU (conocida generalmente como la "GPL") en 1989. La mayoría del sistema GNU, excepto por el kernel, se completó aproximadamente al mismo tiempo. En 1991, Linus Torvalds liberó el kernel Linux bajo los términos de la GPL, creando un sistema GNU completo y operacional, el sistema operativo GNU / Linux (generalmente referido simplemente como Linux).

Las motivaciones políticas y morales de Richard Stallman le han convertido en una figura controversial. Muchos programadores de influencia que se encuentran de acuerdo con el concepto de compartir el código, difieren con las posturas morales, filosofía personal o el lenguaje que utiliza Stallman para describir sus posiciones. Un resultado de estas disputas condujo al establecimiento de una alternativa al movimiento de Software Libre, el movimiento de código abierto.

Stallman ha recibido numerosos premios y reconocimientos por su trabajo, entre ellos una membresía en la MacArthur Foundation en 1990, el Grace Hopper Award de la Association for Computing Machinery en 1991 por su trabajo en el editor Emacs original, un doctorado honorario del Royal Institute of Technology de Suecia en 1996, el Pioneer award de la Electronic Frontier Foundation en 1998, el Yuki Rubinski memorial award en 1999, y el Takeda award en 2001.

1.2.2 Free Software Foundation (FSF)

La Fundación para el Software Libre (FSF) está dedicada a eliminar las restricciones sobre el copiado, redistribución, entendimiento, y modificación de programas de computadoras. La idea es promocionar el desarrollo y uso del software libre en todas las áreas de la computación, pero muy particularmente, ayudando a desarrollar el sistema operativo GNU.

Muchas organizaciones distribuyen cualquier software libre que esté disponible. En cambio, la Fundación para el Software Libre se concentra en desarrollar nuevo software libre y en hacer de este software un sistema coherente, el cual puede eliminar la necesidad de uso del software propietario.

Además de desarrollar GNU, FSF distribuye copias de software GNU, manuales por un costo de distribución, y acepta donaciones deducibles de impuestos (en los Estados Unidos), para apoyar el desarrollo de software GNU. Muchos de los fondos de la FSF provienen de los servicios de distribución.

La Fundación para el Software Libre es una organización sin ánimo de lucro exenta de impuestos que consigue y proporciona fondos para trabajar en el Proyecto GNU.

1.3 Proyectos de Software Libre

Al tomar este tema de los proyectos es evidente que el software libre existe por iniciativas personales de algunos autores o bien como fruto de proyectos bien definidos.

A continuación se mencionaran algunos de ellos por su nivel de importancia hoy en la actualidad.

1.3.1 GNU

El proyecto GNU se inició desde 1984 teniendo como objetivo el desarrollo de un sistema operativo libre completo, similar a UNIX. Después de 15 años, luego de haber adoptado al kernel Linux, el proyecto GNU ha iniciado a dar sus frutos. No en vano, gracias al proyecto GNU y a todos quienes han contribuido al desarrollo de Linux, se estima que existen más de 10 millones de usuarios del sistema operativo GNU/Linux, normalmente denominado en forma impropia como Linux. La primera versión de prueba del sistema GNU utilizó su kernel Hurd, lo que ocurrió en agosto de 1996. Debido a varios años de esfuerzo, el proyecto Hurd continúa, con la esperanza de que su arquitectura superior haga a los sistemas operativos más poderosos, como lo anuncia la FSF.

Un pequeño proyecto derivado de GNU es GNUish, que tiene como objetivo proporcionar un ambiente similar a GNU para pequeños sistemas corriendo DOS y OS/2, aunque también es posible utilizar parte del software GNUish en ambientes MS-Windows. El software que forma parte del proyecto GNUish consiste de software que ha sido compilado para sistemas Microsoft, así como también software que se ha desarrollado como reemplazo a software GNU que no ha sido compilado a los sistemas indicados. Los proyectos GNU, Hurd y GNUish son de la FSF.

1.3.2 GNU/Linux

GNU/Linux es el sistema operativo de mayor éxito en la historia del software libre. El sistema GNU/Linux está conformado por software GNU y por el kernel Linux, el trabajo de investigación que un día de 1991 comenzó a desarrollar Linus Torvalds, un estudiante de la Universidad de Helsinki. Linux estuvo inspirado en Minix, un pequeño sistema operativo UNIX desarrollado por Andy Tannenbaum. Hoy el sistema GNU/Linux es el proyecto de software que está revolucionando al mundo.

Si bien en sus inicios Linux fue el kernel del sistema operativo, ahora hablar de Linux, o mejor, de GNU/Linux, es hablar de un sistema operativo que dispone de cientos de programas libres, la mayoría de ellos desarrollados por el proyecto GNU de la Free Software Foundation. GNU/Linux es un sistema operativo multiusuario y multitarea que corre en muchas plataformas, siendo considerado como uno de los mejores y más eficientes sistemas operativos para computadores personales. No por causalidad el gobierno mexicano, mediante un proyecto que ya está en marcha, se decidió a dotar a 140.000 escuelas con este sistema operativo.

Técnicamente hablando GNU/Linux dispone de casi todo, pero aún le hacen falta más aplicaciones y la facilidad de uso, que necesitan los usuarios comunes y corrientes, que se empieza a alcanzar con la aparición y crecimiento de los dos principales proyectos de software libre que desarrollan un conjunto de aplicaciones de escritorio para ambiente gráfico, como son KDE y GNOME. Si bien GNU/Linux aún no ha alcanzado su completa madurez, el respaldo que viene recibiendo de los grandes productores de hardware y software, de la talla de Intel, IBM, Corel, Oracle y Sun, por nombrar sólo algunos, son claros indicios de que para GNU/Linux y para el movimiento del software libre los mejores tiempos están por venir.

GNU/Linux se distribuye bajo la Licencia Pública General de la FSF (GNU Public License) en forma de distribuciones. Hasta el momento, Red Hat es la distribución de GNU/Linux de mayor éxito.

1.3.3 Debian GNU/Linux

Debian es el proyecto de software libre GNU/Linux, que gracias a sus cerca de 200 desarrolladores voluntarios, combina e integra el software disponible en Internet para formar un sistema operativo completo, de alta calidad y 100% libre. La importancia de Debian radica en su compromiso de dar prioridad a sus usuarios y al software libre, sin que por ello se opongan a su comercialización y sin pedir ningún cargo para ellos. Estos compromisos están plasmados en su Contrato Social Debian.

1.3.4 Sistemas operativos BSD

La Universidad de California en Berkeley es la gestora del desarrollo de varios años del sistema operativo BSD (Berkeley Software Distributions), software de excelencia técnica, que junto al UNIX AT&T de los Laboratorios Bell, se constituye como uno de los dos pilares o versiones de UNIX de los cuales se derivan muchas de las variantes de UNIX de la actualidad. Todo esto fue posible gracias al Grupo de Investigación en Ciencias de la Computación (CSRG) de la Universidad de California, que luego de algunos inconvenientes legales con los dueños de la licencia UNIX, entregó en 1990 el sistema operativo completamente libre BSD-Lite, del cual se derivan los sistemas FreeBSD, OpenBSD y NetBSD.

1.3.5 X Window System, XFree 86

X Window es un sistema gráfico de ventanas gratuito desarrollado por el MIT (Massachusetts Institute of Technology). MIT colocó los archivos fuentes a disposición del público y hoy en día es el estándar de facto para los sistemas de ventanas de muchos sistemas operativos. Casi todas las interfaces gráficas de los sistemas operativos UNIX están basadas en X Window.

Un trabajo derivado del proyecto X Window System es el proyecto XFree 86, Inc, que produce XFree 86, una implementación libremente redistribuible del Sistema X Window que corre sobre GNU/Linux, sistemas operativos tipo UNIX y OS/2. El proyecto XFree86 tradicionalmente ha centrado su trabajo en plataformas basadas en x86 (PCs con procesadores Intel), de donde proviene su nombre, aunque la última versión también soporta otras plataformas, lo cual es uno de sus objetivos actuales.

1.3.6 GNOME y KDE

GNOME y KDE son dos proyectos diferentes con el mismo objetivo, desarrollar un conjunto de aplicaciones en ambiente gráfico con facilidad de uso e interoperatividad, para sistemas operativos UNIX. GNOME es la versión GNU para Linux, en cambio KDE se desarrolla adicionalmente para otros UNIX. Tanto GNOME como KDE constan de diversos programas que permiten intercambio de información entre ellos y dan respuesta a las necesidades de software de cualquier usuario común.

1.3.7 Mozilla

Netscape Corporation basó su trabajo en el navegador libre NCSA's Mosaic hasta alcanzar la cima. Por razones de supervivencia e inspirada en el artículo "La Catedral y El Bazar" de Eric Raimond, donde se contrasta los modelos de desarrollo de software existentes. En mayo de 1998 Netscape causa conmoción en el mundo del software comercial, al liberar el código fuente de su navegador bajo el proyecto de software libre Mozilla, organización que trabaja en el desarrollo de una nueva generación del navegador y software de comunicaciones de Netscape. Este evento fue histórico para Internet ya que Netscape llegó a ser la primera gran compañía de software propietario en dar a conocer archivos fuentes, normalmente considerados como secreto comercial.

1.3.8 Apache

Apache es el servidor HTTP (de páginas Web) más utilizado a nivel mundial. El proyecto de software libre Apache está basado en el servidor HTTP creado por Rob McCool del NCSA (National Center for Supercomputing Applications) de la Universidad de Illinois y es desarrollado por el Apache Group, un grupo de voluntarios unidos a través de Internet, fundado en 1995, debido a que el desarrollo del servidor HTTP del NCSA se detuvo por el retiro de su autor del NCSA a mediados de 1994.

1.3.9 Perl

Perl es uno de los lenguajes de programación más utilizado en Internet que se utiliza para producir contenidos dinámicos en páginas Web. Perl es un lenguaje versátil que combina características del lenguaje C, de los intérpretes de patrones sed y awk y del intérprete de comandos sh. Perl fue creado por Larry Wall en 1986.

1.3.10 Php

Php (Hypertext Preprocessor) es un lenguaje de programación de contenido dinámico que trabaja del lado del servidor. Es un proyecto del apache software foundation

1.3.11 Mysql

Es la base de datos de código abierto más popular sobre el internet, con más de 5 millones de instalaciones activas, ofrece un rápido manejo de información, incluso compite en tiempo de respuesta contra cualquier manejador de base de datos comercial

1.4 Tipos de Licenciamiento para el Software Libre

1.4.1 Licencia Pública General GNU

La Licencia Pública General GNU, llamada comúnmente GNU GPL, es usada para la mayoría de programas GNU, y por más de la mitad de las aplicaciones de software libre.

La licencia GPL (General Public License) es la forma legal mediante la cual el proyecto GNU consigue que un programa posea Copyleft. El proyecto GNU la emplea en la mayoría del software que distribuye. Software GNU es software que es liberado bajo los auspicios del proyecto GNU. La mayor parte del software GNU tiene Copyleft pero no todo, sin embargo todo el software GNU es software libre. El proyecto GNU tiene como meta desarrollar el sistema GNU, que es un sistema operativo completamente libre estilo UNIX.

1.4.2 Licencia Pública General Reducida GNU

No todo el software libre del proyecto GNU se distribuye bajo la licencia GPL porque en algunos casos han considerado necesario que el software libre se pueda incorporar en software propietario, para ello se definió la LGPL, en principio denominada Library General Public License y ahora renombrada como Lesser General Public License. La LGPL se aplica principalmente a las denominadas bibliotecas (conjuntos de programas, funciones o procedimientos).

La Licencia Pública General Reducida GNU es usada por algunas, pero no todas, las bibliotecas GNU. Esta licencia fue llamada en un principio GPL para bibliotecas pero se cambio el nombre debido a que el nombre antiguo animaba a la gente a emplear esta licencia más de lo debido.

1.4.3 Licencia Libre de Documentación GNU

La Licencia Libre de Documentación GNU es una forma para ser usada en un manual, libro de texto u otro documento que asegure que todo el mundo tiene la libertad de copiarlo y redistribuirlo, con o sin modificaciones, de modo comercial o no comercial.

1.4.4 Licencias de Software Libre que se consideran a fin de la GPL (compatibles)

- La licencia X11
- Licencia Expat

- Licencia de derechos de autor estandar ML de Nueva Jersey
- La Licencia General Cryptix
- La Licencia BSD modificada
- La Licencia de ZLib
- Licencia de la librería de función estándar iMatrix
- Licencia W3C
- Licencia de Base de Datos Barkeley
- Licencia OpenLDAP, versión 2.7
- Licencia de Python 1.6a2 y versiones anteriores
- Licencia de Python 2.0.1 2.1.1 y versiones recientes
- Licencia de Perl
- Licencia Artística Clarificada
- Licencia Artística 2.0
- Licencia Pública Zope versión 2.0
- Licencia de fuente abierta Intel(como la Públicada por OSI)
- Licencia de Netscape Javascript
- Licencia eCos versión 2.0
- Licencia Foro Eiffel versión 2
- Licencia de Vim versión 6.1 o reciente

1.4.5 Licencias de Software Libre incompatibles con GPL

Las siguientes licencias son de software libre pero no son compatibles con la GPL, por no tener vínculo con el proyecto GNU

- Licencia Pública General Affero
- Licencia Pública Arphic
- Licencia Original BSD
- Licencia OpenSSL
- Licencia Libre Academica versión 1.1
- Licencia de software abierto versión 1.0
- Licencia Apache versión 1.0
- Licencia Apache versión 1.1
- Licencia Pública Zope versión 1
- Licencia de xinetd
- Licencia de Python 1.6b1 y recientes
- Licencia vieja de OpenLDAP versión 2.3
- Licencia Pública IBM version1.0
- Licencia Pública Común versión 1.0
- Licencia Phorum versión 1.4
- Licencia Pública del Proyecto LaTeX
- Licencia Pública Mozilla(MPL)

- Licencia de Fuente Abierta Netizen (NOSL) versión 1.0
- Licencia Pública Internase versión 1.0
- Licencia Pública Sun
- Licencia de Fuente Abierta Nokia
- Licencia Pública Netscape (NPL)
- Licencia de Fuente Abierta Jabber versión 1.0
- Licencia de Fuente de Estándares de la Industria Sun
- Licencia Q Publico (QPL) versión 1.0
- Licencia FreeType
- Licencia PHP version3.0
- Licencia Zend versión 2.0
- Licencia Plan 9 de Junio del 2003
- Licencia de Fuente Abierta Apple (APSL) versión 2

1.4.6 Licencias de software no libre

Las siguientes licencias no son de software libre y por consecuencia tampoco con compatibles con la GNU GPL

- Licencia Artística (Original)
- Licencia de Fuente Pública Apple (ASPL) versión 1.0
- Licencia SGI Licencia B versión 1.1
- Licencia de Fuente Comunidad Sun
- Licencia del viejo Plan 9
- Licencia Open Public(esta licencia no es una licencia de software libre ya que cuando se modifica una versión y se Pública hay que enviarla al desarrollador inicial la nueva versión)
- Licencia Pública de la Universidad de Utah(no es un licencia libre ya que no permite la predistribución comercial)
- Licencia Pública eCos versión 1.1(por el mismo motivo de la Open Public)
- Licencia del Código Fuente de Sun Solares
- Licencia del Yast
- Licencias Daniel Bernstein s'
- Licencia Fuente de la comunidad Jahia
- Licencia Microsoft de Fuente Compartida
- Licencia Squeek
- Licencia de Acuerdo de Software Fuente Mejorada Activismo(HESSLA)

1.4.7 Licencias de Documentación Libre

- Licencia de Documentación FreeBSD (FreeBSD Documentation Licencse)
- Licencia de Documentación Común Apple version 1.0(Apple s' Common Documetation License)
- Licencia de Públicación libre versión (Open Públication License)

1.4.8 Licencias de Documentación no Libres

La siguiente son Licencias de documentación que no son calificadas como libres:

- Licencia de Directorio Abierto(The Open Directory License)

1.4.9 ¿Qué es el copyleft?

Copyleft es la forma general de hacer un programa software libre y requiere que todas las modificaciones y versiones extendidas del programa sean también software libre.

El modo más simple de hacer un programa libre es ponerlo a dominio público, o sea sin copyright. Esto permitirá a la gente compartir el programa y sus mejoras, si es lo que desean. Pero esto permitiría a la gente no cooperativa convertir el programa a software propietario. Pueden hacer cambios y distribuir el resultado como un producto propietario. Las personas que reciban el programa en su forma modificada no poseen la libertad que el autor original les dio debido a que el intermediario se la ha retirado.

En el Proyecto GNU, la intención es dar a todos los usuarios la libertad de redistribuir y cambiar software GNU. Si el intermediario puede coartar la libertad, se puede tener muchos usuarios, salvo aquellos que carecen de libertad. Así, en vez de poner software GNU bajo dominio público, lo hacemos ``copyleft". Copyleft significa que cualquiera que redistribuya el software, con o sin cambios, tendrá la libertad de copiarlo, redistribuirlo y cambiarlo. Copyleft garantiza que el usuario mantenga su libertad.

El copyleft da incentivos a otros programadores para unirse al software libre. Software libre importante como el compilador C++ GNU sólo existe gracias a ésto.

El copyleft también ayuda a los programadores que deseen contribuir con mejoras al software libre a obtener un permiso para hacerlo. Estos programadores usualmente trabajan para empresas o universidades que harían casi de todo para obtener más dinero.

Un programador podría contribuir con sus cambios a la comunidad, pero su superior querrá hacer esos cambios en un producto de software propietario. Es entonces cuando se explica al

empleador que es ilegal distribuir la versión mejorada del producto salvo como software libre, el empleador normalmente decide liberarlo como software libre en vez de tirarlo a la basura.

Para hacer un programa copyleft, lo primero es darle un copyright; luego se le añaden cláusulas de distribución, que son un instrumento legal que le da a cualquiera el derecho de usar, modificar y redistribuir el código fuente del programa o de cualquier programa derivado de éste pero sólo si los términos de distribución no se cambian. De este modo, el código y las libertades se vuelven legalmente inseparables.

Los desarrolladores de software propietario emplean el copyright para quitar la libertad a los usuarios, el Proyecto GNU emplea ese copyright para garantizar su libertad. Es por esto que cambiamos el nombre ``copyright'' por ``copyleft.''

Copyleft es un concepto general. En el Proyecto GNU, las cláusulas específicas de distribución que emplea están contenidas en la Licencia Pública General GNU, la Licencia Pública General Reducida GNU y la Licencia Libre de Documentación GNU.

La licencia apropiada está incluida en muchos manuales y en cada distribución de código fuente GNU.

La GNU GPL está diseñada para ser fácilmente empleada en un programa si es el dueño del copyright. No tiene que modificar la GNU GPL para hacer esto, sólo añadir una nota en su programa la cual se refiera correctamente a la GNU GPL. Hay que tener en cuenta que si usa la GNU GPL, debe conservar el texto íntegro de la licencia. Es un todo en conjunto; las copias parciales no están permitidas (de igual modo para la LGPL y FDL).

Empleando los mismos términos de distribución para los diferentes programas hace fácil compartir el código entre ellos. Como todo posee los mismos términos de distribución, no hay necesidad de pensar si los términos son compatibles. La GPL Reducida incluye una cláusula que le permite alterar los términos de distribución de la GPL ordinaria, para así poder copiar código con programas cubiertos por la GPL

1.5 Definiciones de Tipos de Software

Si bien cada programa viene acompañado de una licencia de uso particular, existen diversos aspectos en común entre las licencias que hacen posible su clasificación. De acuerdo a ello, es común encontrar términos tales como software shareware, freeware, de dominio público, o de demostración. A estos nombres hay que agregar software libre y software propietario, términos un tanto desconocidos pero que se usan en medios informáticos. Incluso, algunos autores hablan de software semi-libre.

1.5.1 Software Libre

El software libre es software que viene con autorización para que cualquiera pueda usarlo, copiarlo y distribuirlo, ya sea literal o con modificaciones, gratis o mediante una gratificación. En

particular, esto significa que el código fuente debe estar disponible. ``Si no hay fuente, no es software''. Ésta es una definición simplificada.

Si un programa es libre, entonces es potencialmente incluido en un sistema operativo libre tal como GNU, o sistemas GNU/Linux libres.

Hay muchas maneras diferentes de hacer un programa libre. Algunas de las posibles variaciones son descritas a continuación.

Después de saber que el software libre es evaluado por mucha gente y por instituciones, donde se tiene un control total de las correcciones y mejoras que pudiera tener ese software, se podría incluso argumentar que el software libre es más fiable que el software no libre.

"Software Libre" se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

- La libertad de usar el programa, con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias(libertad 2).
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto.

1.5.2 Software de Fuente Abierta

El término software de ``fuente abierta'' es usado por algunas personas para nombrar al software libre. Es un sinónimo del término ``software libre''.

1.5.3 Software de dominio público

El software de dominio público es software que no está protegido con copyright. Es un caso especial de software libre no protegido con copyleft, que significa que algunas copias o versiones modificadas no pueden ser libres completamente.

Algunas veces la gente utiliza el término ``dominio público'' de una manera imprecisa para decir ``libre'' o ``disponible gratis.'' Sin embargo, ``dominio público'' es un término legal y significa de manera precisa ``sin copyright''. Por claridad, se recomienda el uso de ``dominio público'' para ese significado solamente y el uso de otros términos para transmitir los otros significados.

1.5.4 Software protegido con copyleft

El software protegido con copyleft es software libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando estos redistribuyen o modifican el software. Esto significa que cada copia del software, aun si ha sido modificado, debe ser software libre.

En el Proyecto GNU, se protege mediante copyleft casi todo el software que se escribe, porque el objetivo es dar a cada usuario las libertades que el término ``software libre'' implica.

1.5.5 Software libre no protegido con copyleft

El software libre no protegido con copyleft viene desde el autor con autorización para redistribuir y modificar así como para añadirle restricciones adicionales.

Si un programa es libre pero no protegido con copyleft, entonces algunas copias o versiones modificadas pueden no ser libres completamente. Una compañía de software puede compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como un producto propietario de software.

El Sistema X Window ilustra este caso. El Consorcio X libera X11 con términos de distribución que lo hacen software libre no protegido con copyleft. Se puede obtener una copia que tenga esos términos de distribución y es libre. Sin embargo, hay versiones no libres también, y hay estaciones de trabajo populares y tarjetas gráficas para PC para las cuales versiones no libres son las únicas que funcionan. Por ejemplo, si se utiliza este hardware, X11 no es software libre.

1.5.6 Software abarcado por GPL

La GPL (General Public License/Licencia Pública General) de GNU, es un conjunto específico de términos de distribución para proteger con copyleft a un programa. El Proyecto GNU la utiliza como los términos de distribución para la mayoría del software GNU.

1.5.7 El sistema GNU

El sistema GNU es un sistema operativo libre completo estilo Unix.

Un sistema operativo libre estilo Unix consiste en muchos programas. Se ha estado acumulando componentes para este sistema desde 1984; la primera liberación de prueba de un ``sistema GNU completo'' fue en 1996. Se espera que en los próximos meses o tal vez años este sistema este lo suficientemente maduro para recomendarlo a usuarios ordinarios.

El sistema GNU incluye todo el software GNU, así como muchos otros paquetes tales como el Sistema X Window y TeX que no son software GNU.

Debido a que el propósito de GNU es ser libre, cada componente individual en el sistema GNU tiene que ser software libre. No todos tienen que estar protegidos con copyleft, sin embargo;

cualquier tipo de software libre es legalmente apto de incluirse si ayuda a alcanzar metas técnicas. Podemos hacer uso de software libre no protegido con copyleft como el Sistema X Window.

1.5.8 Software GNU

Software GNU es software que es liberado bajo el auspicio del Proyecto GNU. La mayoría del software GNU está protegido con copyleft, pero no todos; sin embargo, todo el software GNU debe ser software libre.

Algo de software GNU es escrito por el personal de la Fundación para el Software Libre (Free Software Foundation), pero la mayoría del software GNU es aportada por voluntarios. Parte del software aportado está protegido con licencias de derechos de autor por la Fundación para el Software Libre; otra parte está protegido con licencias de derechos de autor por los aportadores que los escribieron.

1.5.9 Software semilibre

El software semilibre es software que no es libre, pero viene con autorización para particulares de usar, copiar, distribuir y modificar (incluyendo la distribución de versiones modificadas) sin fines de lucro.
El software semilibre es mejor que el software propietario, desde el punto de vista en que ya no es tan cerrado y la adición de nuevos módulos o funcionalidades por parte de algún desarrollador es muy viable, pero aún plantea problemas y no podemos usarlo en un sistema operativo libre, debido a que es difícil integrar todo a un esquema de libertad absoluta.

Las restricciones del copyleft están diseñadas para proteger las libertades esenciales de todos los usuarios. La única justificación para cualquier restricción acerca del uso de un programa es prevenir que se agreguen restricciones por parte de otras personas. Los programas semilibres tienen restricciones adicionales, motivados por fines puramente egoístas.

Es imposible incluir software semilibre en un sistema operativo libre. Esto obedece a que los términos de distribución para el sistema operativo libre como un todo es la unión de términos de distribución de todos los programas en él. Agregando un programa semilibre al sistema lo haría un sistema semilibre. Existen 4 razones por las que no queremos que esto suceda:

- Se cree que el software libre debe ser para todos--incluyendo empresas, no solamente para escuelas o para quienes gusten de matar el tiempo. Se invita a las empresas a usar el sistema GNU completo y por lo tanto no se recomienda incluir un programa semilibre en él.
- La distribución comercial de sistemas operativos libres, incluyendo sistemas GNU basados en Linux, es muy importante y los usuarios aprecian poder comprar distribuciones comerciales en CD-ROM. Incluyendo un programa semilibre en un sistema operativo cortaría su distribución comercial en CD-ROM.
- La misma Fundación para el Software Libre es no comercial y se estaría habilitado legalmente para usar un programa semilibre ``internamente''. Pero no se hace, porque esto minaría el esfuerzo para obtener un programa que se pudiera incluir en GNU.

- Si hay un trabajo que necesita hacerse con software, entonces mientras no se tenga un programa libre para hacer el trabajo, el sistema GNU tiene un hueco. Se tiene que decirle a los voluntarios, ``Aún no tenemos un programa para hacer este trabajo en GNU .''Si la FSF misma usa un programa semilibre para hacer el trabajo, echaría abajo lo que dice al respecto, alejaría las ganas para escribir un reemplazo libre.

1.5.10 Software propietario

El software propietario es software que no es libre ni semilibre. Su uso, redistribución o modificación está prohibida, o requiere que se solicite autorización o está tan restringida que no se puede hace libre de un modo efectivo.

La Fundación para el Software Libre sigue la regla de no instalar ningún programa propietario en las computadoras excepto temporalmente para el propósito específico de escribir un reemplazo libre para ese programa (con excepción de lo anterior, no hay excusa posible para instalar un programa propietario).

Por ejemplo, la FSF se sentía justificados al instalar Unix en sus computadoras en los '80s, porque lo estaban usando para escribir un reemplazo libre para Unix. Actualmente, debido que están disponibles sistemas operativos libres, la excusa ya no es aplicable; se ha eliminado todos sus sistemas operativos no libres y cualquier computadora nueva instalada debe ejecutar un sistema operativo completamente libre.

1.5.11 Freeware

El término ``freeware'' no tiene una definición clara aceptada, pero es usada comúnmente para paquetes que permiten la redistribución, pero no la modificación (que su código fuente no está disponible). Estos paquetes no son software libre, por lo tanto no se debe usar ``freeware'' para referirse al software libre.

1.5.12 Shareware

El shareware es software que viene con autorización para la gente de redistribuir copias, pero dice que quien continúe haciendo uso de una copia deberá pagar un cargo por licencia.

El shareware no es software libre, ni siquiera semilibre. Existen dos razones por las que no lo es:

- Para la mayoría del shareware, el código fuente no está disponible; de esta manera, no se modifica el programa en absoluto.
- El shareware no viene con autorización para hacer una copia e instalarlo sin pagar una cantidad por licencia, ni aún para particulares involucrados en actividades sin ánimo de lucro. En la práctica, la gente a menudo hace caso omiso a los términos de distribución y lo hace de todas formas, pero los términos no lo permiten.

1.5.13 Nagware

Es una variante de la distribución shareware que prevé distorsiones visuales o de otro tipo durante la utilización del software. Esto aparece una vez que ha finalizado el periodo de prueba, y se mantiene hasta que se paga por el programa. Son programas que son copiados y distribuidos gratuitamente con el fin de permitir al usuario evaluar las potencialidades del producto de cara a una eventual compra. Existen después algunas variantes que están indicadas en los términos de la licencia. La más difundida es el periodo de evaluación: bien el autor especifica un tiempo durante el cual el programa puede ser utilizado gratuitamente (entre 15 y 30 días generalmente). Caducado este periodo es necesario comprar el software o quitarlo del ordenador. La compra supone el envío de una suma de dinero al autor y el consiguiente registro. A veces al registrar un programa se tiene la posibilidad de recibir gratuitamente las actualizaciones.

1.5.14 Cardware

Los programas así distribuidos son copiados y usados con la única condición de que se envíe una postal al autor. Algunos programadores piden postales temáticas... No está permitida la modificación del código sin la autorización del autor que mantiene sus propios derechos sobre el software.

1.5.15 Photoware

Los programas así distribuidos son copiados y utilizados por cualquiera siempre que el usuario envíe una foto suya al autor del software (u otro tipo de foto dependiendo de las especificaciones de la licencia). No está permitida la modificación del código sin la autorización del autor que mantiene sus derechos sobre el software.

1.5.16 Software Comercial

El software comercial es software que está siendo desarrollado por una entidad que tiene la intención de hacer dinero del uso del software. ``Comercial'' y ``propietario'' ¡no son la misma cosa! La mayoría del software comercial es propietario, pero hay software libre comercial y hay software no libre no comercial.

Por ejemplo, Ada de GNU siempre es distribuida bajo los términos de la GPL de GNU y cada copia es software libre; pero los desarrolladores venden contratos de soporte. Cuando sus vendedores les hablan a los clientes potenciales, algunas veces el cliente dice ``Nos sentiríamos más seguros con un compilador comercial.'' Los vendedores responden, ``Ada de GNU es un compilador comercial; sólo que es software libre.''

Para el proyecto GNU, el énfasis está en otro orden: lo importante es que Ada de GNU es software libre; si es comercial no es una pregunta importante. Sin embargo, el desarrollo adicional de Ada de GNU que resulta del negocio que soporta es definitivamente beneficioso.

1.5.17 Otros (Licence Agreements)

Los programas así distribuidos son utilizables sólo si son respetadas las reglas especificadas en la licencia adjunta por el autor. Cada programador decide sus propias reglas que tienen que ser aceptadas por el usuario.

1.5.18 Resumen de los diferentes tipos de software libre, semilibre y freeware

Características del software libre, semi-libre y freeware (ver tabla T1.2).

Tipo de software / Características de la Licencia	Precio cero, uso ilimitado y redistribución permitida	archivos fuentes disponibles	archivos fuente modificables	Revisión pública de archivos fuentes	Todos los derivados deben ser gratuitos
Binarios libres (Freeware)	sí				
"Bibliotecas" libres	sí	sí			
Fuente Abierta estilo BSD*	sí	sí	sí		
Fuente Abierta estilo Apache*	sí	sí	sí	sí	
Fuente Abierta estilo Linux/GNU**	sí	sí	sí	sí	sí

* estas dos clases corresponden a software parcialmente libre
** el software de dominio público puede convertirse en libre, semi-libre o propietario
Tabla T1.2. Características de Software Libre y Semi libre

Capítulo 2

2 Arquitectura de aplicaciones empresariales en capas basada en el estándar J2EE

2.1 El Software y la Empresa

Hoy en día una empresa, corporativo o cualquier entidad necesita extender sus fronteras, reducir el costo, y minimizar sus tiempos de respuestas de sus servicios a clientes, empleados y proveedores.

Típicamente, las aplicaciones que proveen esos servicios deben combinar sistemas de información empresariales (EIS's) existentes, con nuevas funciones de negocio que proveen servicios a un rango de usuarios. Los servicios necesitan tener:

- Alta disponibilidad, para cumplir con las necesidades de ambiente de negocios global de hoy en día
- Seguridad, para proteger la privacidad de los usuarios y la integridad de la empresa
- Escalable y seguro, para asegurar que las transacciones de negocio son rápidamente procesadas

En muchos casos, los servicios empresariales son simplemente como aplicaciones multi-capas, donde las capas de en medio integran EIS's (Enterprise Information Systems) existentes con las funciones de negocio y datos de los nuevos servicios. Las tecnologías Web que maduran se utilizan para proporcionar el primer nivel de usuarios con fácil acceso a complejidades de negocio, y eliminar o reducir drásticamente la administración de usuarios y entrenamiento.

El término empresa hace referencia a una organización de individuos o entidades, que trabajan juntos para lograr objetivos en común. Las organizaciones se dan en todas medidas y formas, grandes o chicas, con o sin provecho, gubernamentales y no gubernamentales.

Las empresas por lo general tienen necesidades en común, como compartir información y procesamiento, manejo y seguimiento de bienes, planeación de recursos, manejo de clientes, protección de conocimiento de negocio, etc., El término software empresarial es usado colectivamente para referir todo el software que envuelve el soportar estos elementos en común de una empresa(ver Figura F2.1)

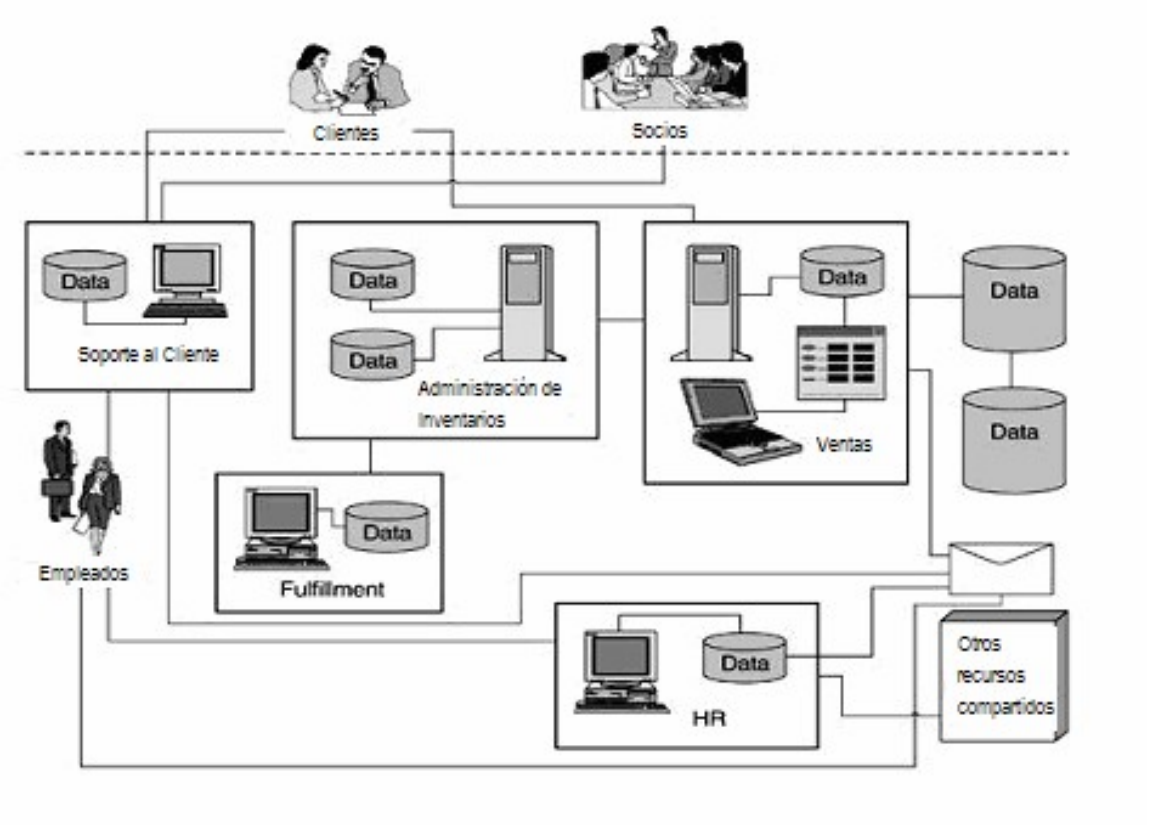

Figura F2.1 Ejemplo estructura de servicio de un software empresarial

En la Figura F2.1 se ve un software empresarial, esencialmente como una colección de diversos sistemas. El software es organizado a través de varias funciones dentro de una organización, por ejemplo, ventas, recursos humanos, etc. Un firewall es proveído para salvaguardar los datos de la empresa de un acceso no autorizado. Algunos sistemas de software son proveídos para interactuar con el manejo de ventas y manejo de inventario, como sea, muchas simplemente son modulos aislado.

El software empresarial está constituido de una multitud de piezas distintas hoy en día, pero las empresas gradualmente han intentado atacar la fuerte necesidad de integrar sus sistemas, para sacar simplemente el mejor provecho de los beneficios de la empresa. Por ejemplo el B2B (business to business, en Internet este término se refiere a poder intercambiar productos, servicios, o información entre negocios) y el B2C (business to customer, es usado para describir la interacción vía Internet entre un negocio y un cliente) son buen ejemplo de dicha integración.

Algunos de los caminos potenciales de la empresa para integrar software empresarial son los siguientes:

- Integrando su soporte a clientes y conociendo un producto en casa, podría proveer nuevos y mejores servicios hacia sus clientes vía la Web.

- Ligando su máquina de mercadeo con el mundo en línea, una empresa podría tener mucho mayor audiencia
- Ligando su manejo de ventas e inventario, una empresa podría concebir algo en específico como bajos costos de ventas por Web para alcanzar una meta de mercado
- Proveyendo una interfaz para uno de los servicios usados por sus empleados, como un sistema de orden de recursos de oficina interno, atándolo a un sistema de cuentas, la empresa puede bajar el costo promedio para mejorar la eficiencia del empleado
- Haciendo el sistema de recursos humanos empresarial disponible en línea, donde podría ser usado como camino para darle a los empleados más control sobre su estado actual y reducir el promedio de costo administrativo para la empresa

2.2 Reto en el Desarrollo de Software Empresarial

Las empresas tienden a crecer cada vez más, contratar más personal, tener más clientes y empiezan a tener más accesos en sus portales, tienen mayores ventas y ganancias, crecen las sucursales en número, etc. Para soportar ese crecimiento, el software empresarial debe de ser escalable en términos de alojar empresas grandes junto con sus operaciones.

Las empresas encuentran limitantes en su crecimiento. Una limitante en común es la inhabilidad del hardware, debido a que la escala de procesamiento se necesita incrementar. Otra limitante es cuando el personal crece y es ya muy difícil acomodar al personal en el mismo lugar. Entonces el verdadero reto de distribución llega, donde múltiples máquinas resuelven las necesidades de procesamiento pero se introduce el réto de tener software distribuido. Con nuevos edificios o nuevos lugares geográficos, se introduce el reto de brindar el mismo nivel de servicio para una empresa localizada ya en varias partes.

Los sistemas legados (legacy) son típicamente diseñados con propósitos específicos en mente y no fueron concebidos para integrarse con otros sistemas en mente. Por ejemplo, la administración del recurso humano fue tratada quizás como una necesidad clara sin mucha interacción con el manejo financiero, manejo de ventas, o con apoyo del cliente. Esto a menudo resulta en arquitecturas de software que son difíciles de integrar.

2.3 Evolución del software empresarial

No hace mucho tiempo, los mainframes gobernaban el mundo informático, y todo el software fue atado a esta entidad central. Las ventajas de un enfoque tan centralizado incluyeron la sencillez de tratar con un sólo sistema para toda necesidad, colocación de todos los recursos. Como desventaja, tenían que lidear con limitaciones físicas de escalabilidad, tenían puntos muy específicos de falla, la accesibilidad remota era limitada, etc.

Como aplicación centralizada era comúnmente referida a una aplicación de una sola capa. En software, una capa es principalmente una abstracción (una division logica de una arquitectura o elemento) y su propósito principal es ayudar a entender la arquitectura asociada con una aplicación específica, por otro lado entender el software en niveles claros y lógicos.

Desde una perspectiva de aplicación, el aspecto problemático más significativo de una sola capa de aplicación fue el entremezclado de presentación, lógica de negocios, y datos. Por ejemplo, si se intenta cambiar algunos aspectos del sistema, en una aplicación de un sólo nivel, todos los aspectos son más confusos; esto es, el lado de la presentación del software esta atado a la lógica de negocio, y la lógica de negocio esta íntimamente ligada a las estructuras de datos. Entonces cualquier cambio significa el afectar potencialmente todo, y habría que considerar muchas cosas. Otro problema que se presenta son las limitaciones de reusar la lógica de negocio o las capacidades de acceso a datos.

El modelo cliente servidor, se podría decir que alivia algunos de estos problemas, moviendo los aspectos de presentación y algunos otros de lógica de negocio en un nivel separado. Desde una perspectiva de aplicación, la lógica de negocio y presentación sigue estando muy mezclada. Pero la idea de dos niveles (capas) introduce también nuevos problemas, por ejemplo, el reto de actualizar la aplicación en un gran número de clientes, con una gran probabilidad de errores a la hora de actualizar, y donde el costo es verdaderamente alto.

Después de conocer el concepto del modelo cliente-servidor, surge la idea de separar la presentación de la lógica de negocios, y la lógica de negocio de los datos ocultos, el termino n-capas (y no solo 3) es representativo de que el software no está limitado a 3 capas solamente, es decir el poder organizar en capas más profundas de acuerdo a las necesidades que se tenga.

Hay que hacer notar que en el modelo de n-capas no implica que deba tratarse de varias piezas de hardware. Una capa es después de todo, una separación de asuntos dentro del software mismo. Las diferentes capas son lógicamente distintas dentro del software pero pueden físicamente existir en la misma máquina, o estar distribuidos a través de múltiples máquinas.

Algunos de las ventajas y beneficios que ofrece el modelo n-capas son:

- Rápido y potencialmente bajo en costo de desarrollo: las nuevas aplicaciones son desarrolladas rápidamente reutilizar código existente, lógica de negocio ya probada con anterioridad y componentes de acceso a datos.
- El impacto de cambios es aislado: Tan pronto un cambio sea hecho a una capa, las otras no son afectadas
- Cambios son más manejados: es fácil reemplazar una versión de un componente de negocio por otro nuevo residente en la capa de negocio (en uno o mas servidores dedicados) sin tener que reemplazar cientos de aplicaciones cliente alrededor de la empresa, o incluso alrededor del mundo.(ver figura F2.2)

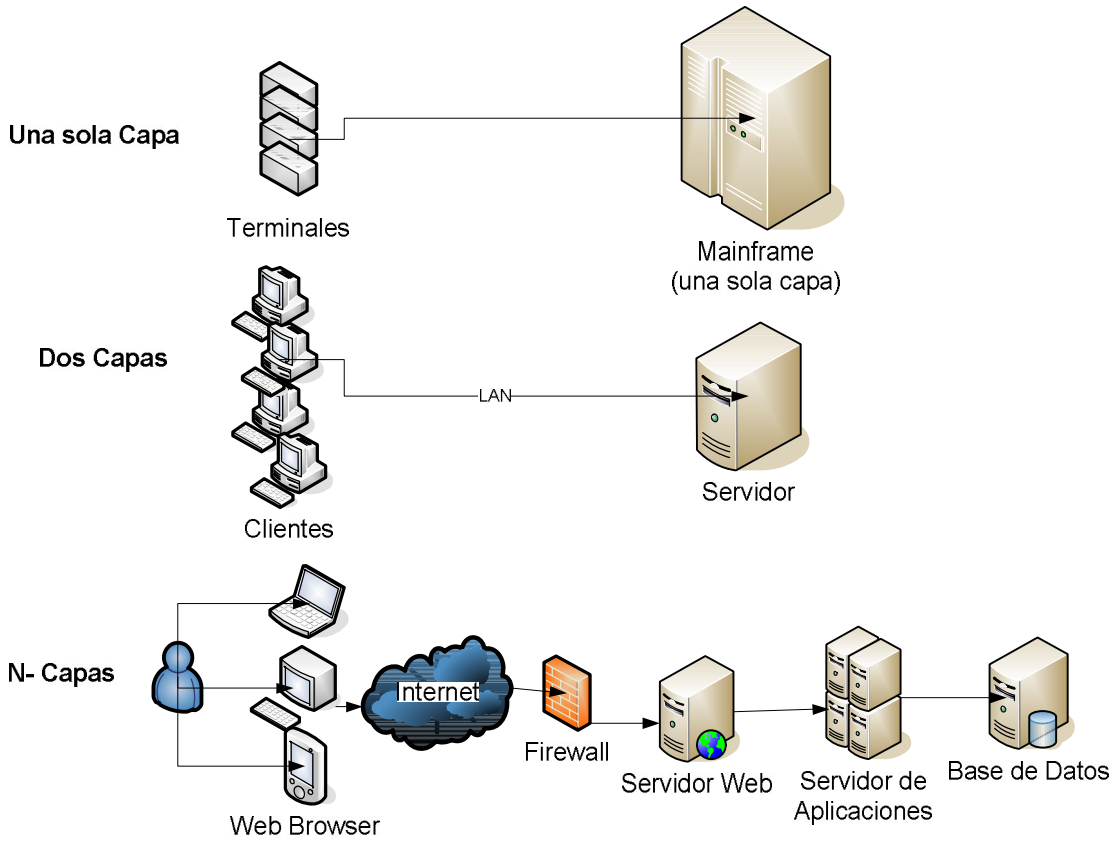

Figura F2.2 Esquema de arquitecturas en capas

2.4 Aspecto Web de las Aplicaciones Empresariales

Pensemos en cualquier proyecto donde se hable acerca de Internet, sobre applicaciones tipo e-business, e-commerce, e-procurement, etc, en donde estas aplicaciones tienen como interfaz de comunicación con el usuario una aplicación Web.

Independientemente de la funcionalidad final de la aplicación, existen una serie de necesidades esenciales al medio que estamos utilizando, y que surgen en cualquier momento. Este es uno de los mayores problemas, es decir, que no surja la necesidad hasta que la aplicación sea demasiado grande y el cambio demasiado costoso. Esto provocara una disminución en la calidad del servicio que se trata de ofrecer.

Una empresa se beneficia de las ventajas de comunicación que ofrece Internet ofreciendo servicios útiles a sus clientes, proveedores, agentes comerciales y usuarios en general tanto externos como internos, pero cuando creamos este nuevo canal debemos de prestarle la atención que requiere porque de ello depende la imagen que los usuarios van a percibir de nuestra empresa.

La principal ventaja de los servicios Web es que automatizan los procesos de servicio que ya tienen las empresas.

2.4.1 Breve historia del Internet

Cuando la Ex Unión de Repúblicas Socialistas Soviéticas (URSS) lanza el primer satélite artificial, en respuesta, los Estados Unidos de Norte América (US), la Agencia de Proyectos de Investigación Avanzada (ARPA) junto con el departamento de la defensa (DoD), establecen un liderazgo en ciencia y tecnología aplicable a lo militar.

Para 1962, Paul Barand, de la corporación RAND (una agencia del gobierno), fue comisionado por la Fuerza Aérea de los Estados Unidos para hacer un estudio sobre cómo mantener el control y mando sobre sus misiles y bombas, después de un ataque nuclear. Esto tenía que ser una Red de Investigación Militar, que pudiera sobrevivir a un ataque nuclear, no centralizada, así si alguna de las ciudades de los Estados Unidos era atacada, los militares podrían incluso tener el control sobre la armas nucleares para un contra ataque.

Barand finalizó con un documento describiendo los caminos que había que seguir para lograr esa finalidad. Su propósito final fue una red de cambio de paquetes de información (packet switched network).

"…el envío de paquetes direccionados es la ruptura de datos para generar diagramas o paquetes que son etiquetados indicando el origen y destino de la información y el envóo de ese paquete de una PC a otra PC de manera que esta información llegue a la computadora final. Esto fue crucial para la realización de una red de computadoras, si los paquetes se pierden en algún punto del viaje, el mensaje debe ser reenviado desde el origen… "

Para 1968, ARPA concedió a ARPANET contratar a BBN. La BBN seleccionó a la mini computadora Honeywell como la base de donde ellos construirían el cambio. La red física fue construida en 1969, enlazando 4 nodos: la Universidad de California en los Ángeles, la SRI (en Standford), la Universidad de California en Santa Bárbara, y la Universidad de Utah, la red fue puesta en funcionamiento vía circuitos con transferencia de 50 kbps.

El primer programa de correo electrónico fue creado por Ray Tomlinson de la BBN. ARPA se había renombrado Agencia de Defensa para Proyectos de Investigación Avanzados (DARPA).

ARPANET estuvo usando el protocolo de control de red o NCP para transferir información, este permitía comunicación entre servidores corriendo sobre la misma red. El desarrollo comenzó con el protocolo después llamado TCP/IP, que fue desarrollado por un grupo encabezado por Vinton Cerf de Stanford y Bob Kahn de DARPA. Este nuevo protocolo, permitía a diversas computadoras en red interconectarse y comunicarse unas con otras.

Un año mas tarde, en 1973 se usó por primera vez el término Internet por Vint Cerf y Bob Kahn en papel sobre el Protocolo de Control de Transmisión.

El Dr. Robert M. Metcalf desarrollo el Ethernet, el cual permitía cable coaxial para mover datos extremadamente rápido. Este fue un componente crucial para el desarrollo de LAN's.

La creación del BITNET por IBM, "Porque es tiempo de red," ("Because it's Time Network"), introdujo la red "guarda y manda." En 1981 La Fundación de Ciencia Nacional creó la columna vertebral, la red CSNET 56 kbps para instituciones sin acceso a ARPANET. Vinton Cerf propuso un plan para una InterConexión de red entre CSNET y la ARPANET.

En 1983, la Tabla de Actividades del Internet (IAB) fue creada. Para el primero de enero, cada máquina conectada a ARPANET tuvo que usar TCP/IP. TCP/IP trajo el núcleo del protocolo de Internet y reemplazó el NCP enteramente.

La Universidad de Wisconsin creó el Sistema de Nombres de Dominio (DNS). Esto permitió que los paquetes fueran direccionados a un nombre de dominio, que sería traducido por el servidor de base de datos en la correspondiente IP numérica. Esto hacía mucho mas fácil el acceso para la gente hacia otros servidores, porque ya no tendrían que recordar números.

En 1992, la sociedad de Internet es alquilada, el World Wide Web(WWW) fue liberado por CERN. Para 1993 un órgano llamado InterNIC servía para proveer servicios específicos de Internet: directorio y base de datos de servicios (por AT&T), registro de servicios (por Network Solutions Inc.), e información de servicios (por General Atomics/CERFnet).

Marc Andreessen, la NCSA y la Universidad de Illinois desarrollaron una interfaz gráfica de usuario para el WWW, llamada "Mosaic para X." Sobre 1994 no se hicieron cambios mayores a las redes físicas de trabajo. La cosa más importante que se hizo fue el crecimiento, muchas nuevas redes fueron agregadas a la columna de NSF. Cientos de miles de nuevos servidores fueron agregados a la INTERNET durante este tiempo de periodo.

La Fundación Nacional de Ciencia anunció que a partir de Abril 30 de 1995, no se podría permitir el acceso directo a la columna NSF. La Fundación Nacional de Ciencia contratada por 4 compañías serían las proveedoras de acceso a la columna NSF. Estas compañías venderían conexiones a grupos, organizaciones, y compañías.

Actualmente la Sociedad de Internet, el grupo que controla la red, está tratando de resolver nuevo TCP/IP para tener disponible billones de direcciones, dada la limitante del día de hoy(se tiene solo un rango limitado de direcciones). El problema que ha surgido es que no se sabe cuando el nuevo y viejo sistema de direcciones estará dispuesto para trabajar, al mismo tiempo que se esté haciendo el periodo de transición.

2.4.2 Breve historia del World Wide Web

En 1945, cuando el Doctor Vannevar Bush, escribió el artículo "As We May Think" para "The Athlantic Online", en que expresaba su preocupación por la cantidad tan grande de información

que existía y estaba siendo generada, el poco tiempo y los ineficientes sistemas que había para encontrarla. Así, y basándose en la tecnología existente en aquel entonces, describió un dispositivo personal, al que llamó "memex", y que imaginaba como un suplemento íntimo a su memoria. Este aparato permitiría a cada individuo almacenar su información en microfilmes, consultarlos rápidamente y, lo que es más importante, crear vínculos entre unos documentos y otros, de modo que durante la lectura de un documento se recordara al lector qué documentos contenían información relacionada. Era una visión de lo que ocurriría sólo 45 años después.

En los años 60, Douglas Engelbart, mientras trabajaba en el Instituto de investigación de Stanford, propuso el NLS (oNLine System), un entorno de trabajo por computadora, con un sistema para almacenar Públicaciones, con catálogos e índices para facilitar la búsqueda, y con reglas establecidas para citar documentos, de modo que fuera más fácil para los lectores acceder a los documentos referenciados. Era un entorno con teclado, pantalla, ratón e impresora, con la posibilidad de teleconferencia y correo electrónico a través de una red de computadoras para una rápida comunicación entre los profesionales. Poseía las herramientas básicas de composición, estudio, organización y modificación de información. Los archivos tenían una organización de tipo jerárquica, se trabajaba con los documentos en modo multiventana, para ver varios documentos a la vez en ventanas diferentes, se podían copiar objetos seleccionados de una ventana a otra.

En si el termino "hipertexto" fue difundido con formalidad por Ted Nelson en 1965, en su articulo "A File Structure for the Complex, the Changing, and the Indeterminate," que leyó sobre la vigésima conferencia anual de la Asociación de Maquinaria Computacional (ACM). Ted Nelson ideo un modelo para la interconexión de documentos electrónicos. El proyecto Xanadu aun continúa luchando para conseguir un modelo de hipertexto superior al que trajo la World Wide Web.
Actualmente el proyecto Xanadu, propone un nuevo modelo (Xanadu model) que permite ser un sistema más profundo que el actual, donde existan:

- Ligas irrompibles
- Copyright simple y suave
- Que origine conexiones
- Dos caminos para las ligas
- Un manejo de versión más profundo
- Incrementar publicidad

Pero, ¿Como hacer todo esto?, pues según este modelo, se propone que se haga de una manera muy simple, hacer contenido disponible con ciertos permisos; después distribuir y mantener documentos simples como lista de esos contenidos, para ser llenado por el navegador(del mismo modo que los navegadores ahora llenan en archivos con formato GIF's). Pareciera ser un modelo interesante de implantación, pero por el momento tenemos el World Wide Web.
La World Wide Web fue inventada en 1989 por un informático del CERN (organización Europea de Investigaron Nuclear) llamado Tim Berners-Lee. Era un sistema de hipertexto para compartir información basado en Internet, concebido originalmente para servir como herramienta de comunicación entre los científicos nucleares del CERN. Tim Berners-Lee había estado experimentando con hipertexto desde 1980, año en el que programó Enquire, un programa para almacenar piezas de información y enlazarlas entre ellas. Enquire se ejecutaba en un entorno multiusuario y permitía acceder a varias personas a los mismos datos. Berners-Lee entrego su

propuesta en 1989 al CERN, en septiembre de 1990 recibió el visto bueno comenzó a escribir el nuevo sistema de hipertexto. Empezó a trabajar sobre un navegador editor GUI de hipertexto usando el ambiente de desarrollo NEXTStep. Tim creó el programa que puso como nombre "World Wide Web ."El propósito original se reformuló con mucho ánimo y Robert Cailliau se unió y fue co-autor de la nueva versión. Para noviembre de ese año, el primer servidor Web fue el nxoc01.cern.ch, llamado info.cen.ch, y la primera página http://nxoc01.cern.ch/hypertext/WWW/TheProject.html.

A finales de 1990 Nicola Pellow se unió y empezó a trabajar sobre un navegador modo línea. El navegador en modo línea y el navegador World Wide Web, demostraron ser capaces de acceder a archivos de hipertexto.
Los documentos necesitaban un formato que fuera adecuado para su misión. La portabilidad de los documentos en esos años era un verdadero problema. Se adoptó el lenguaje html (hipertext mark-up language) y como protocolo de red el http(Hiper Text Tranfer Protocol).

A principios de 1993 había alrededor de 50 servidores. Existían básicamente dos tipos de browsers: el original, gráfico, pero sólo para plataformas NeXT, y el browser en modo de línea, preparado para cualquier plataforma pero muy limitado y muy poco atractivo. En febrero se lanzó la primera versión alfa del navegador "Mosaic for X", desarrollado en el NCSA (National Center for Supercomputing Applications). Funcionaba en X Windows, que era una plataforma popular entre la comunidad científica. En abril el tráfico de la WWW era el 0.1% del total de Internet. El CERN declaraba la WWW como tecnología de acceso gratuito. En septiembre ya había versiones de Mosaic para PC y Macintosh. El tráfico alcanzaba el 1% de todo el tráfico de Internet y había más de 500 servidores. Es el comienzo del crecimiento explosivo de la Web. A finales del 94 ya había más de 10,000 servidores y 10 millones de usuarios. En 1997, más de 650,000 servidores.

Hoy, en 2006, la Web es algo cotidiano para una gran parte de los más de 500 millones de usuarios de Internet que hay en todo el mundo. Sus utilidades son diversas, su impacto en la economía mundial es apreciable. No sólo hay documentos de texto: hay imágenes, vídeos, música, se pueden comprar cosas, en general se mueve mucho dinero sobre ella.

2.5 Protocolos de Internet

2.5.1 Modelo OSI

En 1984, la Organización Internacional de Estandarización (ISO) desarrolló un modelo llamado

OSI (Open Systems Interconnection). El cual es usado para describir el uso de datos entre la conexión física de la red y la aplicación del usuario final. Este modelo es el mejor conocido y el más usado para describir los entornos de red.

Figura F2.3 Capas del modelo OSI

Como se muestra en la figura F2.3, las capas OSI están numeradas de abajo hacia arriba. Las funciones más básicas, como el poner los bits de datos en el cable de la red están en la parte de abajo, mientras las funciones que atienden los detalles de las aplicaciones del usuario están arriba.

En el modelo OSI, el propósito de cada capa es proveer los servicios para la siguiente capa superior, resguardando los detalles de como los servicios son implementados realmente. Las capas son abstraídas de tal manera que cada capa cree que se está comunicando con la capa asociada en la otra computadora, cuando realmente cada capa se comunica sólo con las capas adyacentes de la misma computadora.

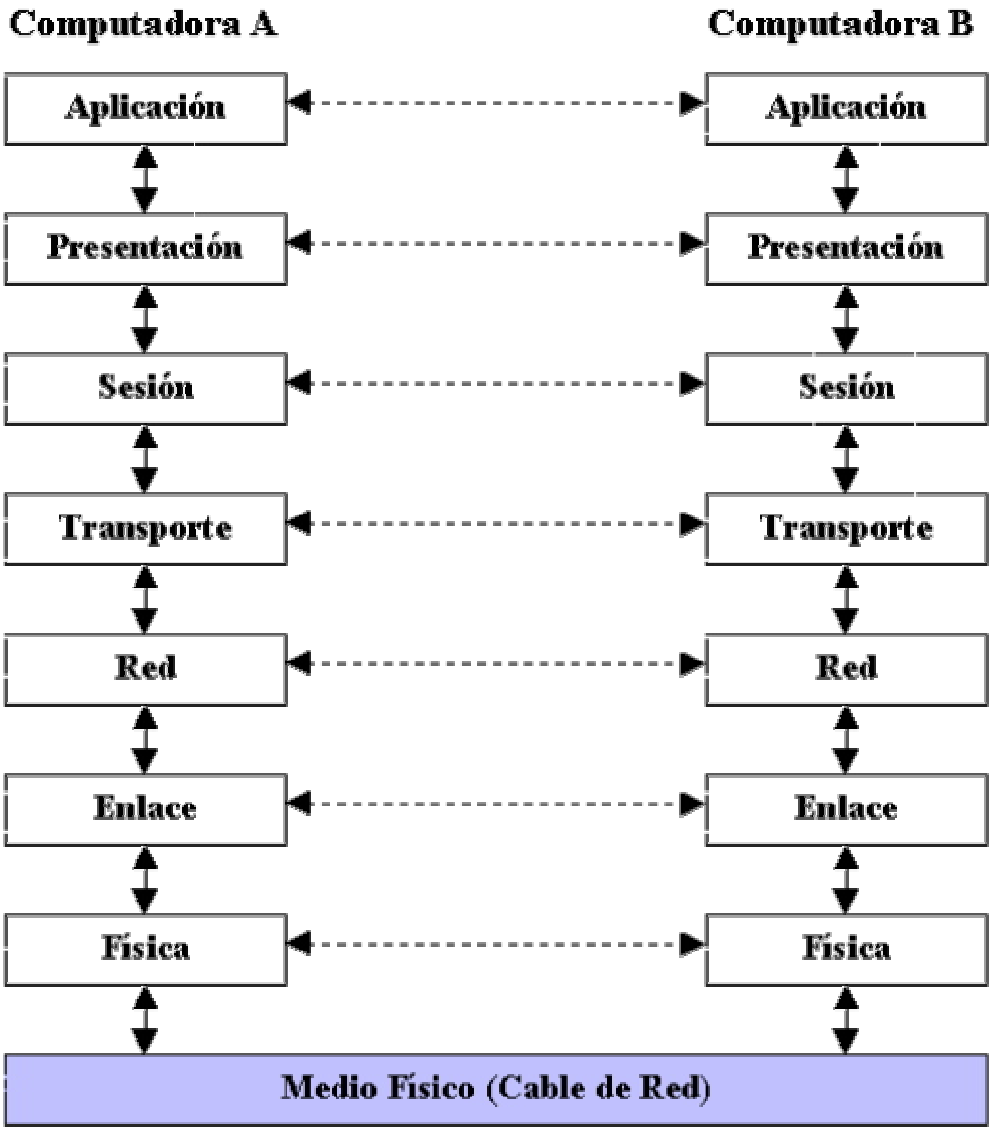

Figura F2.4 Ejemplo de comunicación entre capas

En la figura F2.4 se aprecia que, a excepción de la capa más baja del modelo OSI, ninguna capa puede pasar información directamente a su contraparte en la otra computadora. La información que envía una computadora pasa por todas las capas inferiores, la información entonces se mueve a través del cable de red hacia la computadora que recibe y hacia arriba a través de las capas de esta misma computadora hasta que llega al mismo nivel de la capa que envió la información. Por ejemplo, si la capa de red envía información desde la computadora A, esta información se mueve hacia abajo a través de las capas de Enlace y Física del lado que envía, pasa por el cable de red, y sube por las capas de Física y Enlace del lado del receptor hasta llegar a la capa de red de la computadora B.

La interacción entre las diferentes capas adyacentes se llama interfaz. La interfaz define que servicios la capa inferior ofrece a su capa superior y como esos servicios son accesados. La serie de las reglas que se usan para la comunicación entre las capas se llama protocolo.

2.5.2 Protocolos

La base de Internet, y razón principal de su éxito, son sus protocolos. Dentro de cada capa se utilizan distintas normas o protocolos, llegando incluso a depender, dentro de un nivel, la norma utilizada del servicio a prestar

2.5.3 Nivel de transporte

El protocolo de nivel de transporte original era el Network Control Protocol, NCP, diseñado para ARPANET, funcionó hasta que el sucesivo crecimiento con otras redes dio lugar a ARPA Internet y además provocó que se fuera degradando la fiabilidad extremo a extremo de la red, forzando la necesidad de un nuevo protocolo para el nivel de transporte, el Protocolo de Control de transmisión, TCP, diseñado especialmente para tolerar subredes no fiables.

Es un protocolo orientado a la conexión que queda establecido cuando un nodo determinado comienza a enviar paquetes a otro nodo. Todos los paquetes entre los dos nodos pasan por la misma ruta durante todo el tiempo que dura la conexión. Al tener una ruta fija y única durante el tiempo que dura la conexión si en un momento dado alguno de los enlaces o enrutadores involucrados en formar el circuito virtual tiene algún problema, la conexión entre los nodos origen y destino queda rota.

Los servicios como correo electrónico, transferencia de archivos o acceso remoto, necesitan que los caracteres que se van tecleando en un extremo vayan llegando al otro extremo conservando el orden en que se han introducido, o que el archivo que estamos transfiriendo no pierda o duplique partes del mismo. Se necesita un protocolo que proporcione un flujo de bytes fiable para los dos sentidos de la conexión. Nuestro protocolo es el TCP, que nos garantiza que los bytes que salen del nodo origen son entregados en el nodo destino en el mismo orden y sin duplicados.

Cuando lo que se necesita transmitir es voz o video en tiempo real, es más importante transmitir con una alta velocidad que el garantizar que lleguen absolutamente todos los paquetes, con el orden adecuado y sin duplicados. En esta situación, nuestras necesidades son mejor satisfechas por el protocolo de nivel de transporte llamado Protocolo de Datagramas de Usuario, UDP, que se caracteriza por ser un protocolo no orientado a conexión, es decir, puede que algunos de los paquetes enviados con este protocolo no lleguen nunca, lo hagan varias veces o lleguen en desorden. Cada paquete lleva suficiente información como para alcanzar el destino, reencaminándose el flujo en el caso de que falle algún nodo o enlace. Entre los inconvenientes, simplemente recordar que no se está a salvo de pérdidas, repeticiones y desordenes de los paquetes, por lo que los procesos que usen este protocolo pueden tener una carga adicional de trabajo.

2.5.4 Nivel de Inter-red

A principios de los ochenta se introdujo un nuevo protocolo de nivel de Inter-red, el Protocolo de Internet, IP. Se trata de un protocolo no orientado a conexión, encargado de las cuestiones relativas a direccionamiento de los paquetes que le suministra la capa de transporte.

De esta forma, el protocolo que principalmente se identifica con Internet es el Transmission Control Protocol / Internet Protocol, TCP/IP. Si bien la parte fundamental de la estructura en la que se basan todas las aplicaciones, es la establecida por la norma IP, encargado de determinar los procedimientos de direccionamiento y encaminamiento que deben seguir todas las informaciones transmitidas, independientemente de la red física que se utilice para la conexión.

Como cada servicio tiene sus propias necesidades, existen diferentes protocolos de niveles superiores que usan IP. Aunque el protocolo IP establece las normas para que los paquetes alcancen su destino, lo que no se garantiza es cuándo lo van a alcanzar, cuántos o en qué orden, es decir, ofrece un servicio no orientado a conexión.

2.5.5 Direcciones IP

Entre los conceptos aportados por el protocolo IP están las denominadas direcciones IP, encargadas de identificar de manera única cada máquina o nodo dentro de Internet.

Las direcciones Internet son números de 32 bits, es decir, cubre desde 0 a 2^32, aunque en lugar de usarse un espacio de direcciones plano del tipo: 1,2,3,... se eligió establecer una estructura en las direcciones, de forma que una dirección IP consta de cuatro números separados por puntos. Como para la representación de cada número se han destinado ocho bits, estos pueden ir de 0 a 255, es la denominada notación numérica con puntos. Así, por ejemplo, la dirección Internet "195.76.188.1 "se corresponde de forma única con un nodo dentro de la red, de forma que todo paquete que lleve este destino sólo acabará su viaje felizmente si llega a él.

Se establecieron cinco clases de direcciones IP, denominadas clase A, B, C, D y E. La forma más sencilla de diferenciarlas es mirando el primer número de la dirección.
Se puede ver las distintas clases, su estructura y rangos(ver tabla T2.1). Se detallan los campos dedicados a identificador de red (redID), identificador de host u ordenador (hostID) e identificador de grupo de multidifusión (multicasting group ID, en inglés).

Clase	Rango
A	0.0.0.0 a 127.255.255.255
B	128.0.0.0 a 191.255.255.255
C	192.0.0.0 a 223.255.255.255
D	224.0.0.0 a 239.255.255.255
E	240.0.0.0 a 247.255.255.255

Tabla T2.1 Clases de direcciones IP

Este formato de dirección facilita que se puedan aplicar máscaras que permitan diferenciar direcciones de nuestra red con direcciones fuera de ese ámbito, así como posibles tratamientos de la misma mediante rápidas comparaciones a nivel de bit. Estamos ante lo que se denominan subredes (subnet en inglés), establecidas por al administrador de cada red en concreto, con objeto de facilitar las labores de enrutamiento, disminuyéndose el tamaño de las tablas de encaminamiento intermedias.

2.5.6 Usuarios y dominios

Las direcciones IP son tratadas por los diversos nodos por donde el paquete atraviesa, sin embargo existe un modo alternativo de direccionamiento utilizando el concepto de dominio como alias de una dirección IP. A cada usuario en Internet se le asocia una dirección Internet única, formada por el identificador de usuario y el identificador del ordenador o dominio en que se encuentra, separados ambos por el carácter "@". La sintaxis general de cualquier dirección Internet es : USERID@DOMINIO.

Hay que tener especial cuidado con la distinción entre mayúsculas y minúsculas, dado que se consideran letras distintas y, por tanto, direcciones distintas, así como con la no presencia de espacios en blanco dentro de la dirección. Las distintas partes que forman el dominio reciben el nombre de subdominios. El subdominio más a la derecha es el de carácter más general, denominándose dominio de nivel alto.

Existen dos tipos de dominios de nivel más alto aunque en algunos países se ha definido también un subdominio que les permita diferenciar el tipo de organización, como por ejemplo "ac" para instituciones académicas o "co" para las organizaciones comerciales.

Dominios de organización: se basan en el plan de direccionamiento creado antes de que Internet fuese una red internacional. Contienen definiciones del tipo de organización a la que pertenece la computadora (educativa, comercial, militar, etc.). Cuando Internet se expandió fue necesario definir nuevos dominios de nivel alto que cubrieran esta situación (ver tabla T2.2).

DOMINIOS DE TIPO DE ORGANIZACIÓN	
DOMINIO	SIGNIFICADO
com	Organización comercial
edu	Institución educativa
gov	Institución gubernamental
int	Organización internacional
mil	Organización militar

net	Organización de red
org	Organización sin ánimo de lucro

Tabla T2 .2 Clasificación de dominios

Por ejemplo: En la dirección ayuda@nodo50.org, el identificador de usuario es "ayuda "y el dominio "nodo50.org ." Su dominio de nivel más alto, "org ," nos dice que el servidor pertenece a una organización sin ánimo de lucro. Por último, el subdominio que está más a la izquierda (a la derecha de la @), nos dice el nombre concreto del ordenador que utiliza ese usuario, "nodo50 ." (ver tabla T2.3)

DOMINIOS DE NIVEL ALTO GEOGRÁFICOS			
DOMINIO	SIGNIFICADO	DOMINIO	SIGNIFICADO
aq	Antártida	is	Islandia
ar	Argentina	it	Italia
at	Austria	jp	Japón
au	Australia	kr	Corea del Sur (Korea)
be	Bélgica	kw	Kuwait
bg	Bulgaria	li	Liechtenstein
br	Brasil	lt	Lituania
ca	Canadá	lu	Luxemburgo
ch	Suiza (Cantones Helvéticos)	lv	Latvia
cu	Cuba	mx	México
cn	China	my	Malasia (Malaysia)
cr	Costa Rica	ni	Holanda
de	Alemania (Deutschland)	no	Noruega
dk	Dinamarca (Denmark)	nz	Nueva Zelanda
ec	Ecuador	pl	Polonia
ee	Estonia	pr	Puerto Rico
eg	Egipto	pt	Portugal
es	España	re	Reunión

fi	Finlandia	se	Suecia
fr	Francia	sg	Singapur
gb	Gran Bretaña	si	Eslovenia (Slovenia)
gr	Grecia	th	Tailandia (Thailand)
hk	Hong Kong	tn	Túnez
hr	Croacia	tw	Taiwan
hu	Hungría	uk	Reino Unido (United Kingdom)
ie	República de Irlanda	us	Estados Unidos (United States)
il	Israel	ve	Venezuela
in	India	za	Sudáfrica

Tabla T2.3 Dominios geográficos: dominios de nivel alto que definen la localización geográfica.

2.5.7 Sistema de nombres de dominio

Las direcciones que valen son las IP, debido a que cuando utilizamos direcciones de dominio, necesitamos disponer de un servicio denominado Sistema de Nombre de Dominios (Domain Name System, DNS), que es un servicio TCP/IP que se encarga de establecer las correspondencias entre los nombres de dominios y sus correspondientes direcciones IP. Gracias al servicio de DNS, son equivalentes las direcciones : ayuda@nodo50.org y ayuda@195.76.188.2

Al ser un servicio, para utilizarlo necesitamos indicar a nuestros programas dónde encontrar al servidor correspondiente, información que el proveedor de acceso a Internet. Normalmente, para conexiones realizadas por vía telefónica, la dirección IP que se asigna a nuestra máquina es dinámica, es decir, en el proceso de conexión, el servidor de nuestro proveedor nos asigna una dirección IP temporalmente, válida durante esa conexión concreta y que cambiará la próxima vez que nos conectemos. Sin embargo, nuestra dirección Internet no se ve alterada. El servidor DNS de nuestro proveedor se preocupará de traducir en cada ocasión la parte de dominio de nuestra dirección Internet a la dirección IP que tengamos asignada en ese momento.

La asignación de los números IP no se hace por el capricho de cada usuario, sino que es el Centro de Información de la Red Internet (InterNIC) delegado el encargado de tomar estas decisiones.

Estos organismos toman las decisiones relativas a los identificadores de red (redID), mientras que la parte de la dirección dedicada a las computadoras o hosts depende del administrador de cada red, quien tomará igualmente las decisiones relativas a la creación de subredes.

Para registrar un dominio o un número IP, lo habitual es que se encargue de ello el proveedor de acceso a Internet, aunque siempre existe la posibilidad de rellenar una serie de formularios necesarios para registrarse.

2.5.8 Números de puerto

¿Cómo podemos diferenciar a qué servicio se está refiriendo un cliente que se dirige a una dirección IP dada?.

La respuesta viene a través del número de puerto (PORT NUMBER). Se trata de un concepto proveniente del mundo UNIX, nuevamente. Tanto TCP como UDP identifican una aplicación mediante un número de puerto de 16 bits. Así, los servidores, que son aplicaciones, tienen asignado igualmente un número de puerto. Existe un número de puerto típico para cada servicio, de forma que se dice que una aplicación servidora está "escuchando "en el puerto que se le ha asignado, su puerto típico normalmente, a la espera de peticiones de clientes. Cuando éstas llegan, se atienden, bien de forma secuencial o bien de forma concurrente.

Los números de puerto típicos son asignados por la Autoridad Internet de Números Asignados (Assigned Numbers Internet Authority, IANA) y están entre 1 y 1023. Algunos ejemplos de puertos son: puerto 21 para FTP o 23 para TELNET.

Las aplicaciones clientes también usan números de puerto en nuestra máquina, son aplicaciones usando TCP también, sólo que no son puertos específicos sino que sólo necesitan ser únicos mientras los use el cliente, por ello se denominan puertos efímeros. Igualmente, hay procesos del sistema que usan puertos, denominados puertos reservados.

La forma en que se establece el puerto al que nos dirigimos, caso de no ser el puerto por defecto, consiste en especificarlo tras la dirección IP, separado por dos puntos, por ejemplo: 195.76.188.2:21. En caso de tener que utilizar algún número de puerto en especial este deberá sernos comunicado por el proveedor del servicio.

Para acceder a cada servicio de Internet, necesitamos un programa capaz de interpretar los estándares de cada servicio de Internet. La evolución de Internet, junto con el desarrollo del servicio Web, han provocado que los programas que se usan para acceder a páginas Web, los denominados navegadores o exploradores, asuman cada vez más funciones, o lo que es lo mismo, sean capaces de acceder a más servicios, de forma que nos permiten realizar transferencias de archivos, uso del correo o participación en grupos de noticias.

2.5.9 Nivel de red/enlace

En los protocolos usados en Internet, según nos acercamos al medio físico, la diversidad de los mismos provoca que existan varios protocolos a nivel de red/enlace para adaptarse a las peculiaridades de cada medio físico.

Un usuario conectándose por una línea serie, tenemos la posibilidad de que se trata de una línea de la red telefónica conmutada (RTC) o una línea punto a punto. Ambos casos fueron

contemplados, definiéndose estándares para cada uno de ellos. Así, se definió el protocolo de Internet para líneas serie (Serial Line Internet Protocol, SLIP) y el protocolo para líneas punto a punto (Point to Point Protocol, PPP) destinados a implementar la funcionalidad del nivel de red y enlace sobre los citados medios físicos.

Si bien el protocolo SLIP está específicamente diseñado para el transporte de tráfico TCP/IP, la tendencia actual es hacia el uso cada vez mayor del protocolo PPP, ya que, aunque su nombre pueda despistarnos, también es apto para líneas telefónicas conmutadas, las normales en nuestra casa u oficina, siempre que nuestro proveedor de Internet disponga de un servidor PPP para atender nuestra llamada.

El protocolo PPP posee algunas características que lo hacen más interesante:

- Negociación de la configuración: al utilizar SLIP, es necesario conocer tanto nuestra dirección IP como la de nuestro proveedor, lo que causa problemas es el caso de que este asigne dinámicamente las direcciones. Igualmente, existe la posibilidad de tener que configurar algunos parámetros un tanto "oscuros ," como la máxima unidad de transmisión (MTU), máxima unidad de recepción (MRU), el uso de cabeceras de compresión, etc. Todos estos pasos se simplifican notablemente con el protocolo PPP gracias a mecanismos de negociación durante la conexión.
- Login automático: casi todos los programas SLIP/PPP llaman y hacen login de forma automática, siguiendo un archivo de comandos. Sin embargo, es conveniente que el sistema del proveedor de servicio envíe prompts estándar de cara a facilitar el proceso. Por ejemplo, debería enviar un "login : "cuando espere que nuestro sistema la envíe nuestro identificador y "password : "para pedir la clave. Algo que no siempre ocurre, teniéndose que recurrir entonces a escribir un archivo de conexión específico (script) o realizarla de forma manual. PPP considera la posibilidad de que se utilicen dos posibles métodos de automatización, el protocolo de autentificación de claves, (Password Authentication Protocol, PAP) y el protocolo de autentificación por Challenge-Handshake, (Challenge-Handshake Authentication Protocol, CHAP). Ambos aportan un mecanismo para enviar la pareja login/clave de forma transparente al sistema remoto.
- Capacidad de transporte multiprotocolo: al ser PPP más reciente se le ha dotado de una mayor potencia, aunque a efectos de conectarse a Internet, donde sólo se usa el protocolo TCP/IP no es significativa..

2.6 Internet, Intranet y Extranet

El termino "Intranet "sugiere ambigüedades porque además suele confundirse con Internet. La confusión de su existencia es con la World Wide Web o simplemente Web, pues Internet se localiza sobre redes físicas y técnicas mientras que la Web lo hace sobre contenidos accesibles ubicados sobre esa infraestructura física y técnica. Hoy en día el término Intranet esta definido de diferentes maneras, pero el significado de todos llegan a un mismo sentido de equivalencia, entonces, una Intranet es un conjunto de contenido compartido por un grupo bien definido dentro

de una organización. Una Extranet es un conjunto de contenido compartido bien definido pero que atraviesa limites empresarios.

Estas diferencias de acceso son muy importantes porque el contenido Web usa la misma infraestructura técnica independientemente de las decisiones de acceso. Los términos Intranet y Extranet adquieren hoy sentido como dos conceptos complementarios para brindar acceso a determinados contenidos Estos términos van evidentemente a evolucionar pero por hoy, un conjunto de contenido accedido solo por miembros de una organización es una Intranet, aun en el caso de que la información atraviese la infraestructura pública Internet

2.7 Concepto de desarrollo empresarial basado en Web

Si no se tiene claro desde el principio cual es el objetivo, no se puede discernir cual será el mejor camino. Uno de los cambios fundamentales que se ha producido, en el desarrollo de software en los últimos años, está más cercano a la filosofía que a la tecnología, es decir, es el cambio de orientación hacia el servicio.

Una aplicación Web es una aplicación orientada al servicio. Mientras en las aplicaciones tradicionales el objetivo es el procesamiento de datos, en las aplicaciones Web el objetivo es ofrecer un servicio a cada uno de los usuarios que se conectan.

Mientras en una aplicación de gestión tradicional el único beneficiario es la empresa y por tanto, la interfaz está concebida para satisfacer las necesidades de ese beneficiario; en una aplicación Web el beneficiario debe ser cada uno de los usuarios que se conecta a la misma. Cada usuario tiene una visión de lo que espera de ese servicio, su beneficio.

Muchas veces de los casos es un error trasladar el análisis de la aplicación de gestión interna de la empresa a una aplicación Web. No vamos a pensar que vamos a poder dictar las normas y nuestros usuarios las seguirán firmemente. La Web es un sistema basado en la oferta y la demanda, si no ofreces el servicio que demandan tus usuarios, simplemente no lo utilizan, y la empresa deja de percibir los beneficios que esperaba sacar de esta utilización.

Lo mencionado hasta este punto va a marcar el análisis, las tecnologías a utilizar, el diseño, el uso, y todo lo que se necesita para poder proveer ese servicio incluidos los sistemas sobre los que se ejecuta la aplicación, los técnicos encargados de dar soporte, los sistemas de gestión de los que se alimenta la aplicación Web, etc.

Cuando hablamos de los requisitos de una aplicación Web y tratamos de definir el concepto, inmediatamente queremos identificarlo con paginas Web. Las páginas Web han sido concebidas para Públicar información a toda la comunidad Internet de forma sencilla. Esta simplicidad se vuelve en contra cuando queremos desarrollar aplicaciones complejas en entornos empresariales.

Cualquier conjunto de páginas Web que interactúen con el usuario, ofreciéndole la información solicitada y recogiendo datos del mismo, llegaría a percibirse como una aplicación Web por parte de ese usuario, pero desde el punto de vista de empresa, el enfoque cambia.

Una aplicación Web genérica, con independencia de la implementación concreta en la que estemos pensando para nuestra empresa, es una plataforma desde la que se ofrezca servicios a terceros, y aunque tengamos claras las necesidades inmediatas y los servicios que queremos ofrecer a nuestros clientes, representantes, etc., en primer momento van a ser los usuarios los que nos indican qué y cómo quieren consumir ese servicio. Y es cuando nuestra aplicación es capaz de adaptarse a las nuevas demandas, que es lo que queremos construir.

La decisión de que servicios ofrece a sus usuarios corresponde a la empresa y no está limitada por la tecnología que esta utilizando. Muchos intentos de ofrecer servicios por Internet han fracasado por estas carencias, en donde en parte se le ha culpado a Internet, a que la mayoría de los clientes no están conectados, a que los usuarios no saben utilizar la plataforma.

Internet es un medio de comunicación, no se podría culpar al mensajero del contenido del mensaje. Si tenemos un buen contenido y se ajusta a la demanda de nuestros clientes la utilización, la utilización del medio es cuestión de tiempo, y de capacitación en el mejor de los casos.

Pensando en una aplicación Web empresarial no como un programa informático sino como una plataforma de integración de servicios, existen requisitos a cumplir y que son nombrados a continuación:

- Control de acceso por usuarios
- Acceso en línea a los datos
- Capacidad de integración
- Independencia del diseño
- Flexibilidad para la ampliación
- Administrable al 100% por la empresa
- Calidad de servicio asegurada

Estos puntos están desarrollados dentro de los estándares marcados por el Internet para asegurar el acceso desde cualquier lugar y desde cualquier dispositivo con acceso a Internet.

2.7.1 Control de acceso por usuarios

Al ingresar a la página principal del sitio, existe una casilla donde nos pide usuario y otra donde pide contraseña. En otras nos muestra información general, y cuando nos registramos, el sistema reconoce el perfil y actúa en consecuencia mostrándonos información particular del tipo: si tenemos correo nuevo o noticias de nuestro interés.

Un perfil de usuario esta compuesto por una serie de roles que la aplicación conoce y que tienen un comportamiento distinto según los roles de los que disponga el usuario conectado.

Enfocándonos en una aplicación empresarial, esto es de vital importancia, porque no queremos mostrar la misma información a los mismos usuarios, incluso diferentes clientes pudieran tener diferentes restricciones, o tal vez tarifas, etc.

Todas las restricciones que se necesiten y que irán saliendo según la aplicación Web vaya dando más y más servicios estan implementadas mediante una arquitectura que ofrezca estos mecanismos de base. Con código abierto es posible hacer casi cualquier cosa, el problema radica en que todos esos parches que introducimos en páginas sueltas terminan debilitando la aplicación, haciéndola vulnerable y difícil de mantener.

Un histórico de los usuarios conectados es importante para conocer el uso que se hace del sistema. La gestión de usuarios y sus perfiles es una parte importante de nuestro sistema y deberíamos tener herramientas que nos ayuden en estas tareas.

Los componentes que manejan el control de acceso suelen utilizar base de datos para almacenar información de usuarios y sus perfiles, pero es aconsejable que la arquitectura fuera abierta de forma que permitiera la utilización de otro tipo de repositorios como archivos XML, texto plano, directorios LDAP, etc.

El control de acceso no sólo es la autenticación del usuario, también debe incluir los componentes para el comportamiento selectivo de la aplicación según el perfil del usuario. El objetivo se resume en tratar de forma personalizada a un cliente, que incluso ha entrado a formar parte de la organización, con sus respectivos privilegios.

2.7.2 Acceso Online a los datos

Una vez que tenemos identificada que tipo de información se quiere suministrar o recoger de los usuarios de ese servicio una duda que siempre surge cuando se ataca un proyecto de este tipo es: ¿dónde deben estar los datos?. En realidad esta pregunta depende de otras dos: ¿dónde están los datos?(lugar lógico de almacenamiento), y ¿qué tiempo de vida tienen?

Si la información que está pidiendo el usuario tiene un tiempo de vida corto solo se ofrecen desde donde se produce, es decir que si la información es replicada a alguna otr aparte temporalmente para ser accesados esos datos no existirán, más que donde se crearon de origen. De cualquier otra forma nuestro servicio no tendría la calidad esperada por el cliente y se estaría dando un mal servicio, que en muchos casos es peor que no darlo.

Como regla para aplicar a la hora de tomar decisiones pensamos que toda la información que vamos a dar a través de Internet debe tener el mismo nivel de consistencia que si nos atendieran personalmente. Si no se ofrece este requisito es mejor no darlo, porque en el momento que un usuario desconfié de la información obtenida a través de Internet volverán a llamar por teléfono dando por muerto el esfuerzo que hayamos realizado por automatizar el servicio, eso sin contar con la desconfianza de que no se sabe si el sistema ha funcionado mal. Por tanto la aplicación es capaz de atacar a los sistemas de gestión de la empresa para acceder o dejar informacion en línea. Recordemos que aunque no estemos pensando en, por ejemplo, hacer reserva de

mercancías en un primer momento, el sistema debe ser capaz de implementarlo en el momento que lo necesitemos.

2.7.3 Capacidad de gestión

Una aplicación Web en principio no sustituye a los sistemas que ya tiene la empresa, por el contrario, es el envoltorio que los transforma en servicio. Para ello se cuenta con una alta capacidad de integración con los sistemas existentes: Bases de Datos, ERP's (Enterprise Resource Planning) y otro sistemas automáticos.

Para esta comunicación es necesario utilizar protocolos estándar para conectar con cualquier recurso que necesite la aplicación. Siguiendo un esquema de desarrollo bajo plataforma Java, independientemente del fabricante del servidor de bases de datos nuestra aplicación se conecta a la base de datos a través de un protocolo JDBC.

El servidor de correo es accesible por pop3, imap, smtp. Si necesitamos un servicio de mensajería este se accede mediante JMS (Java Message Service) ya que éste es parte del estándar J2EE.

La utilización de protocolos estándar es lo que nos da la independencia de elegir quéproducto del mercado se ajusta mejor a nuestras necesidades y de cambiar, a otra base de datos "más potente ,"cuando el sistema lo requiera sin necesidad de modificar la aplicación.

Algo que es muy importante recalcar es que hay que tener mucho cuidado a la hora de elegir determinada tecnología ya que existen tecnologías como Active-x, DCOM, etc. Las cuales casan nuestra aplicación con protocolos propietarios que no hayan alcanzado el estatus de estándar y con ello quedarnos enganchados al fabricante.

Una aplicación Web extiende los sistemas que ya dispone la empresa, haciendo una labor integradora entre los mismos.

2.7.4 Flexibilidad de Ampliación y cambio

Las aplicaciones Web ofrecen servicios y los servicios cambian o crecen por nuevos requerimientos del mercado. La única forma de evaluar la flexibilidad de una aplicación es analizar su arquitectura, y esto lleva meses. La arquitectura separa la lógica de negocio del diseño grafico, Si se quiere cambiar el diseño no se tenga que rescribir la aplicación, un ejemplo es seguir las Arquitecturas MVC (Model View Controller) que mas adelante se hablará al respecto.

2.7.5 Administrable

Una aplicación Web provee de herramientas de administración, de los servicios sobre los que se sustenta: como con las bases de datos. De lo contrario se restringe a si misma al intentar ampliar funcionalidades.

2.7.6 La importancia de los servicios base

Internet ha cambiado el concepto del software, un programa funciona correctamente cuando llega al usuario, cuando se convierte en servicio, y esto no termina en la máquina que se está ejecutando. Todas las capas que cubren a una aplicación hasta convertirla en servicio son otros servicios que deben estar funcionando correctamente.

Y todo esto está montado sobre una plataforma que asegure:

- Alta disponibilidad
- Actualización continua
- Seguridad

Pero vamos a relacionar lo temas pasados, de forma mas resumida podemos decir que el HTTP es un protocolo pensado para Públicar páginas estáticas, no se creópara desarrollar aplicaciones. En 1989, en el CERN, Tim Berners-Lee propuso compartir la información utilizando documentos de texto de hiperenlaces. Junto con Anders Berglud desarrollaron una versión de hipertexto llamada lenguaje de marcado de hipertexto (HTML). Tim denominóa su sistema de hipertexto la World Wide Web.

Su sencillez, sin duda, ha sido la base de su éxito, la posibilidad de crear páginas con información formateada desde un simple editor de textos y sin la necesidad de tener conocimientos profundos en lenguajes de marcado, unido a la proliferación de herramientas para la creación de páginas Web han inundado Internet de páginas formateadas en HTML.

Para visualizar estas páginas se crearon los navegadores Web. Sencillos programas, en un principio, que traducían el lenguaje de marcado HTML a un formato visual. Las compañías que desarrollaron estos navegadores han ido incorporando extensiones al lenguaje para ir disminuyendo sus carencias y para satisfacer las demandas que le solicitaba el mercado.

Los navegadores Web acceden a las páginas HTML conectándose a servidores Web mediante el protocolo HTTP (Hipertext Transfer Protocol), o protocolo de transferencia de hipertexto. Este protocolo provee de la semántica necesaria para la recuperación de recursos, como páginas HTML, utilizando URLs (Unit Resource Locator) que son direcciones universales para la localización de recursos en el mundo Internet.

Por un lado tenemos una forma de pedir información que se encuentre en cualquier lugar de Internet, o más exactamente hacer llegar la petición de información al servidor Web que se supone que la tiene (HTTP).

Por otro lado tenemos un lenguaje que reconocen todos los navegadores Internet, y que son capaces de transformarlo en un formato con diseño, por tanto sabemos que si la información solicitada al servidor nos es devuelta en HTML podrá ser visualizada en un cliente universal, independiente del fabricante y del dispositivo.

Parece lógico pensar que si se formatea cualquier tipo de información que tengamos en nuestros sistemas de gestión en documentos HTML la tendremos accesible desde cualquier lugar y desde cualquier dispositivo que soporte HTML.

La tarea no es sencilla y las tecnologías han ido cambiando para responder más eficazmente a este principio. Mientras unas tecnologías basaban su diseño en la sencillez de uso y rápido aprendizaje del lenguaje como el PHP, otras han invertido su esfuerzo en crear una arquitectura robusta y escalable para dar soporte a complejas aplicaciones corporativas como es el caso de la J2SE y J2EE(Java 2).

2.7.7 Navegador Web o Browser

El navegador se considera como una interfaz de usuario universal. Dentro de sus funciones están la petición de las páginas Web, la representación adecuada de sus contenidos y la gestión de los posibles errores que se puedan producir. Para todo esto, los fabricantes de navegadores les han dotado de posibilidades de ejecución de programas de tipo script, con modelos de objetos que permiten manipular los contenidos de los documentos. Estos lenguajes de programación son VBScript, JScript (ambas de Microsoft) y JavaScript (de Netscape), y proporcionan las soluciones llamadas del lado del cliente, client side y permiten realizar validaciones de datos recogidos en las páginas antes de enviarlos al servidor y proporcionan un alto grado de interacción con el usuario dentro del documento.

Otras de las posibilidades de los navegadores es la gestión del llamado HTML dinámico (Dinamic HTML, DHTML). Éste está compuesto de HTML, hojas de estilo en cascada, (Cascade Style Sheets, CSS), modelo de objetos y scripts de programación que permiten formatear y posicionar correctamente los distintos elementos HTML de las páginas Web, permitiendo un mayor control sobre la visualización de las páginas.

En esta línea, los navegadores han ido un poco más allá y permiten la visualización de documentos XML (eXtensible Markup Language) después de haber sido transformado adecuadamente a HTML por las hojas de estilo extensibles (eXtensible Style Sheets, XSL). De esta manera se elige visualizar ciertos elementos y otros no, dependiendo de las circunstancias.
Además, los navegadores permiten la ejecución de aplicaciones dentro de los documentos mostrados. Las dos posibilidades más populares son la tecnología ActiveX y los applets Java que se descargan del servidor Web y se ejecutan en la JVM (Java Virtual Machine) del navegador.

2.8 Software Empresarial y Software basado en componentes

Cuando el software orientado a objetos llegó a la escena de desarrollo de software, se esperó extensamente que la adopción de técnicas de desarrollo de software orientado a objetos fomentaría el uso, pero esto sólo fue parcialmente realizado. Una de las razones de este parcial éxito fue la fina diseminación de objetos y la dificultad oculta de lograr rehusar a gran escala ese nivel de objetos.

Un componente de software es diseñado en otro nivel de abstracción y provee una función completa o un servicio. Usando interfaces los componentes están expuestos, ellos son combinados entre si rápidamente para construir aplicaciones de gran tamaño rápidamente y sean mas efectivas en cuanto a costo.

Modelos de componentes distribuidos han sido desarrollados para dirigir software basado en componente en el contexto de software distribuido de la empresa. Los Modelos de componentes esencialmente proveen un "sistema operativo "para distribuir su desarrollo de software basado en componente, ejemplos de esto son los Sun Enterprise JavaBeans (EJB) que son parte de la plataforma J2EE.

El software empresarial ha experimentado una evolución gradual tratando de proporcionar el valor más grande de la empresa, el software empresarial encara algunos retos distintos. Esto incluye entre otros, escalabilidad, distribución, seguridad, y la necesidad de trabajar con diversos conjuntos de tecnologías. Varias arquitecturas han evolucionado y han intentado alcanzar a cubrir esos retos.

Un software basado en componentes, requiere por supuesto que los componentes de diferentes fuentes sean compatibles. Varios modelos de componentes han sido desarrollados a través de los años para proveer un conocimiento en común, Microsoft ActiveX, después COM, y SUN Microsystem Applets y JavaBeans son ejemplos de dichos modelos de componentes.

2.8.1 Desarrollo de sistemas con UML

EL Unified Modeling Language (UML) es un lenguaje gráfico para el modelado y desarrollo de sistemas. Este permite modelado y soporte de visualización para todas las fases de desarrollo de un software, desde su análisis de requerimientos hasta la especificación, construcción y despliegue.

El UML tiene sus raíces en varias notaciones orientadas a objetos, el más prominente entre ellos son las notaciones popularizadas por Booch, Rumbaugh, y Jacobson. Así, aunque el UML se ha formalizado apenas hace unos pocos años, sus antecesores se han usado para diseñar y especificar sistemas software-intensivos desde principios de la década de los 90

La diferencia de notación y metodología por lo regular es motivo de confusión. El UML es una notación que es aplicada usando muchos enfoques distintos. Estos enfoques son las metodologías.

La unión de las notaciones vinieron a mediados de los 90's. A principios de 1997, varios consorcios enviaron respuestas a Object Management Group (OMG) sobre su petición para proponer un meta-modelo común para describir los sistemas de software. Un consorcio encabezado por Rational Software (ahora propiedad de IBM) mandó la especificación UML 1.0. Este incorporó las características delanteras de varios modelando las anotaciones inclusive esos de Booch, de Rumbaugh, y de Jacobson. En la petición de la OMG, muchos de los consorcios que

competían cooperaron con el grupo liderado por Rational para redefinir el UML 1.0 en UML 1.1, que fue aceptado por la OMG a finales de 1997.

UML continúa evolucionando bajo la dirección de la OMG. Por ejemplo, recientemente propuso extensiones que proveen notaciones para modelado de datos, modelado de aplicaciones Web, y mapeo de construcciones J2EE a UML.

El UML tiene el amplio apoyo de la industria. En virtud de que es la especificación sostenida por los 850 miembros de la OMG, es el estándar de la industria de software para modelado visual y desplegado. El hecho que todas las herramientas líderes para modelar software-intensivo ahora soporte UML, las hace también ser un estándar.

La idea central detrás de usar UML es el capturar los detalles significativos acerca del sistema, como que el problema esta claramente entendido, o la solución de arquitectura está desarrollada, o una implementación escogida es claramente identificada y construida.

Una rica notación para modelado visual de software facilita este ejercicio. El UML no sólo provee la notación para construcción de bloques básicos, si no también provee caminos para expresar conjunto de relaciones complejas entre bloques de construcción básicos.

Las relaciones son estáticas o dinámicas en naturaleza. Las relaciones estáticas giran principalmente alrededor de los aspectos estructurales de un sistema. Relaciones de herencia entre un par de clases, interfaces implementadas por una clase, y dependencia sobre otra clase.

Ambos aspectos, estáticos y dinámicos de un sistema, son capturados en forma de un diagrama UML. Están organizados en áreas específicas de modelado visual llamadas vistas.

Los siguientes tipos de diagramas son provistos por el UML:

- Diagrama de Clase: Es el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones.
- El mundo real puede verse desde abstracciones diferentes (subjetividad).(ver figura F2.6)

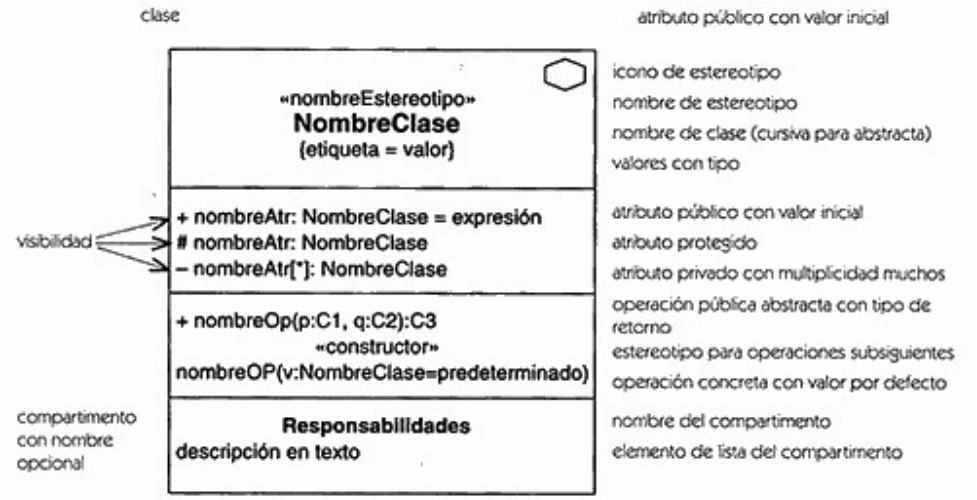

Figura F2.6 Ejemplo diagrama de clase

- Diagrama de Objetos: Provee una vista en forma de foto de las relaciones que existen entre instancias de clases en un punto del tiempo dado. Un diagrama de objetos es útil para capturar e ilustrar, en una forma estática, compleja las relaciones dinámicas dentro del sistema.(ver figura F2.7)

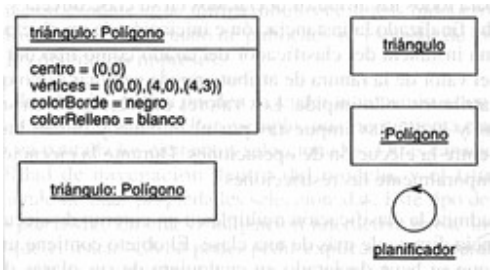

Figura F2.7 Ejemplo diagrama de objetos

- Diagrama de Estado: Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro.

  Los Diagramas de Estado representan autómatas de estados finitos. Son útiles sólo para los objetos con un comportamiento significativo. Cada objeto está en un estado en cierto instante. El estado está caracterizado parcialmente por los valores algunos de los atributos del objeto. El estado en el que se encuentra un objeto determina su comportamiento. Cada objeto sigue el comportamiento descrito en el Diagrama de Estados asociado a su clase. Los Diagramas de Estados y escenarios son complementarios, los Diagramas de Estados son autómatas jerárquicos que permiten expresar concurrencia, sincronización y jerarquías de objetos, son grafos dirigidos y deterministas. La transición entre estados es instantánea y se debe a la ocurrencia de un evento. (ver figura F2.8).

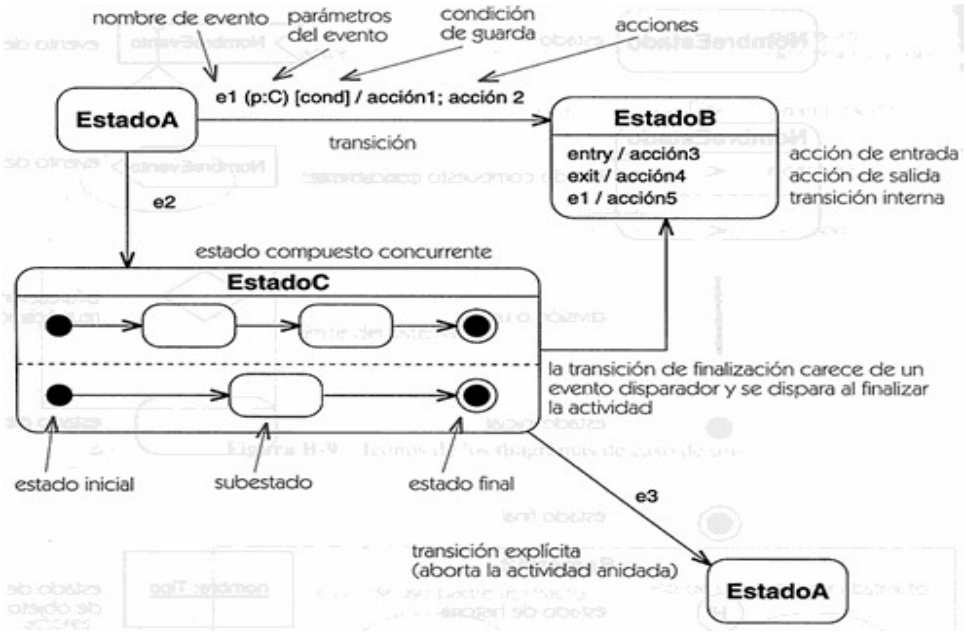

Figura F2.8 Ejemplo diagrama de estado

- Diagrama de actividad: Un diagrama de actividad es una extensión de un diagrama de estado y es similar en concepto a un flujo. Un diagrama de actividad te permite modelar la conducta del sistema en términos de interacción o flujo de control entre distintas actividades u objetos. .(ver figura F2.9)

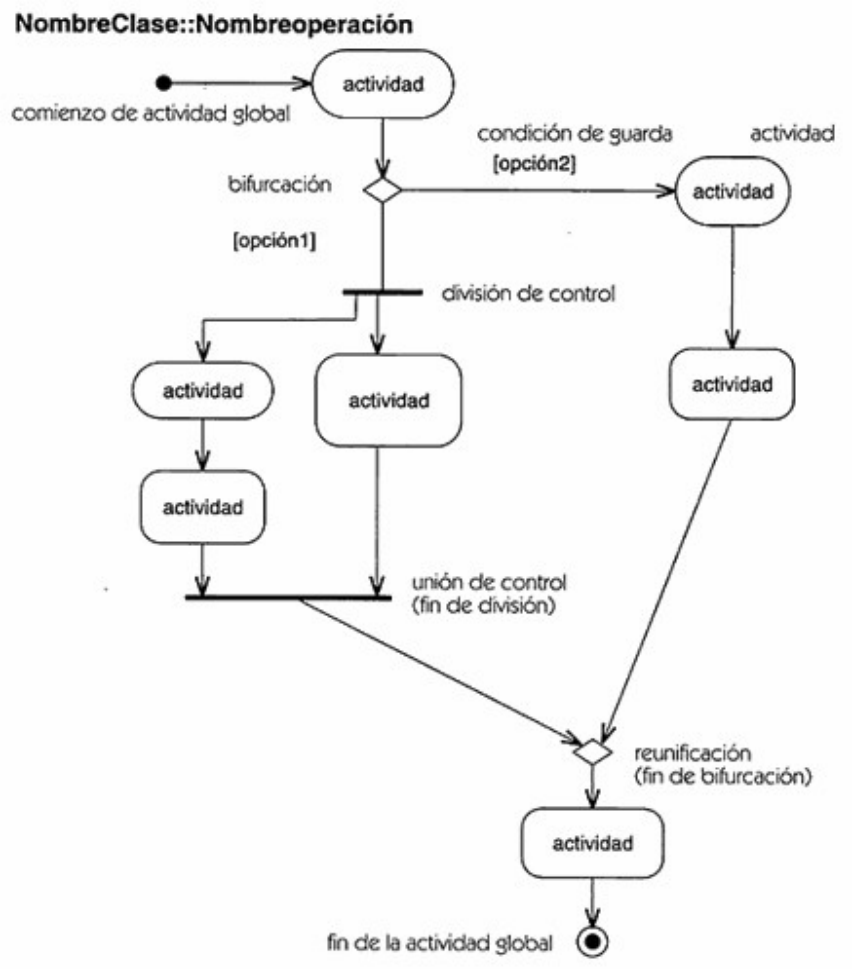

Figura F2.9 Ejemplo diagrama de actividad

- Diagrama de interacción: Los diagramas de interacción son usados en el UML y existen de dos maneras como diagrama de colaboración y el diagrama de secuencia.

- Diagrama de secuencia: Usado para modelado de intercambio de mensajes entre objetos de un sistema. Los diagramas secuencia incluso capturan el tiempo relativo ordenado de mensajes intercambiados. (ver figura F2.10)

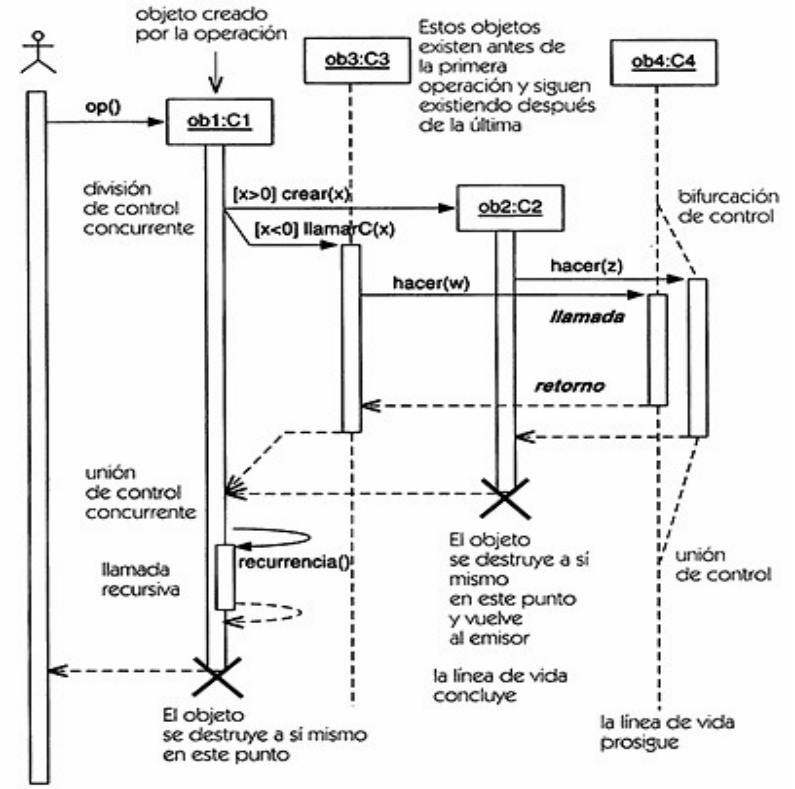

Figura F2.10 Ejemplo de diagrama de secuencias

- Diagrama de colaboración: El intercambio de mensajes se captura en el contexto de las relaciones estructurales generales entre objetos. .(ver figura F2.11).

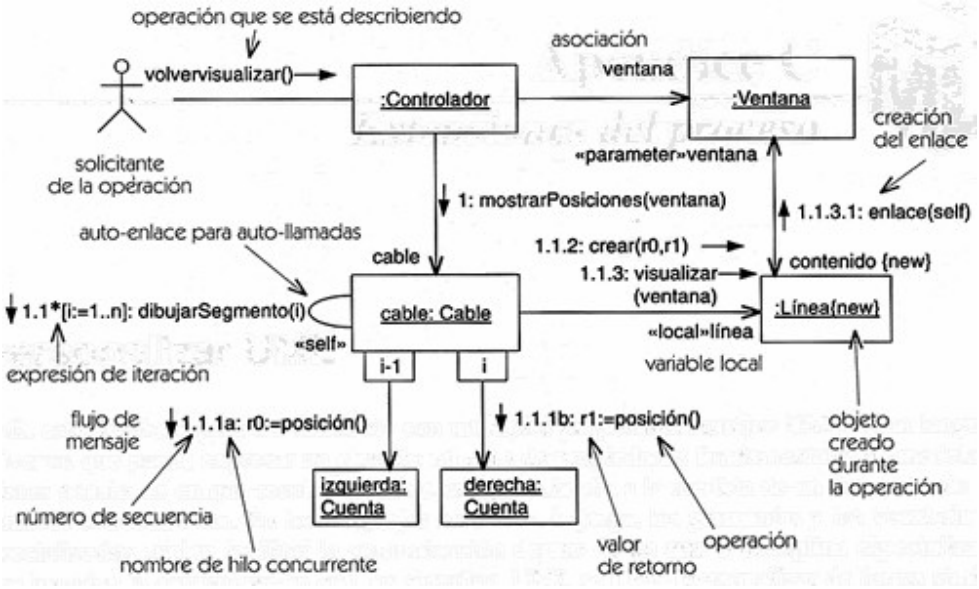

Figura F2.11 Ejemplo de diagrama de colaboración

Los 2 diagramas son equivalentes, y esto es posible convertir de uno a otro. La interacción entre diagramas son comúnmente usados para modelar el flujo de control en un caso de

uso y para describir como los objetos interactúan durante la ejecución de una operación, como la realización de una operación de interfaz.

- Diagrama de componentes: Describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.(ver figura F2.12)

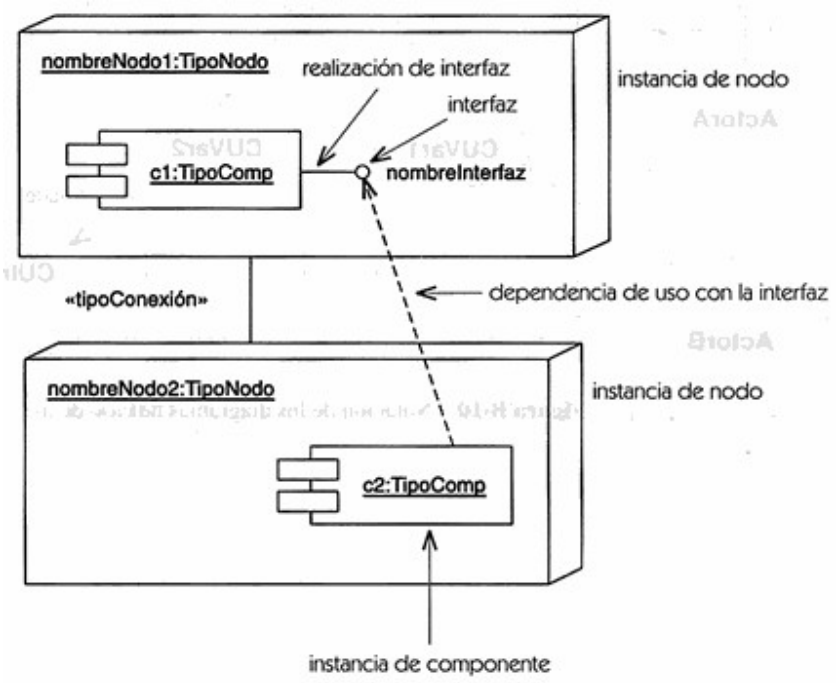

Figura F2.12 Ejemplo de diagrama de componentes

- Diagrama de despliegue: Muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instacias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

- Diagrama de paquetes: Cualquier sistema grande se debe dividir en unidades más pequeñas, de modo que las personas puedan trabajar con una cantidad de información limitada, a la vez y de modo que los equipos de trabajo no interfieran con el trabajo de los otros.

  Los paquetes contienen elementos del modelo al más alto nivel, tales como clases y sus relaciones, máquinas de estado, diagramas de casos de uso, interacciones y colaboraciones; atributos, operaciones, estados, líneas de vida y mensajes están

contenidos en otros elementos y no aparecen como contenido directo de los paquetes. .(ver figura F2.13)

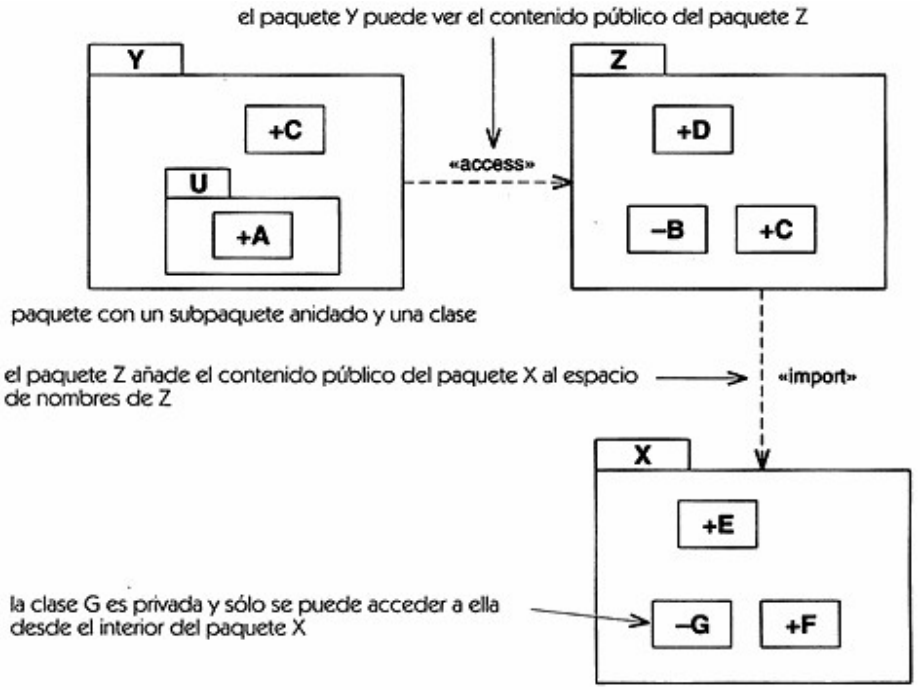

Figura F2.13 Ejemplo de diagrama de paquetes

2.9 Introducción al J2EE

Sun Microsystems ha organizado la plataforma Java 2 en 3 ediciones: Micro Edition (J2ME), Standard Edition (J2SE), Enterprise Edition (J2EE). De estos productos, el J2EE es el más relevante para desarrollo de aplicaciones Java empresariales. Estas ediciones tienen las siguientes caracteristicas:

- La plataforma Java 2, Micro Edition (J2ME): Plataforma para el desarrollo de software para dispositivos embebidos como teléfonos, dispositivos de agenda móvil, etc.
- La plataforma Java 2, Standard Edition (J2SE): La más familiar de las plataformas Java 2. Es incluso conocida como Java Development Kit (JDK) e incluye capacidades como Applets, JavaBeans, etc.
- La plataforma Java 2, Enterprise Edition (J2EE): es una plataforma para desarrollar aplicaciones a escala empresarial. J2EE esta diseñado para ser usado en conjunto con el J2SE, de hecho el J2SE es un subconjunto del J2EE

Pero realmente, ¿que es la plataforma J2EE?, bueno en realidad el J2EE se define como una arquitectura para desarrollo complejo, aplicaciones java empresariales.

J2EE fue originalmente anunciado por Sun Microsystems a mediados de 1999 y fue oficialmente liberado a finales de 1999. El J2EE, es relativamente nuevo, está aun teniendo significantes cambios de versión en versión, especialmente en el área de los Enterprise JavaBeans (EJB).

El J2EE consiste de lo siguiente:

- Diseño de lineamientos para desarrollar aplicaciones empresariales usando J2EE
- Una referencia de implementación para proveer una visión operacional del J2EE
- Una suite de pruebas de compatibilidad para uso de terceras partes para verificar su compatibilidad de productos con el J2EE
- Muchas Interfaces de programación de aplicación (API's) para habilitar acceso genérico a recursos empresariales e infraestructura.
- Tecnologías para simplificar el desarrollo Java empresarial

La plataforma construida sobre Java con el eslogan "Escribe una vez, Corre donde sea "vía un grupo de tecnologías y un conjunto de API's y estas son soportadas y limitadas por 3 elementos clave, llamados la referencia de implementación, el diseño de lineamientos, y la suite de compatibilidad.

La plataforma Java 2, Enterprise Edition (J2EE) reduce el costo y complejidad de desarrollo multicapa, así como de servicios empresariales. Las aplicaciones J2EE pueden ser rápidamente implementadas y mejoradas, respondiendo a los tiempos de respuesta que las empresas necesitan.

J2EE define una arquitectura estándar con los siguientes elementos:

- J2EE Platform.- un estándar de plataforma para hospedar aplicaciones J2EE
- J2EEE Compatibility Test Suite.- una suite de pruebas de compatibilidad para verificar que una plataforma J2EE cumpla con el estandar de la plataforma J2EE
- J2EE Referente Implementation.- Una referencia de implementación para prototipos de aplicaciones J2EE y para proveer una definición operacional de la plataforma J2EE
- J2EE BluePrints.- un conjunto de las mejores prácticas para desarrollo multicapa, servicios cliente.

¿Cómo vino el J2EE a ser tan interesante?. Java, originalmente llamado Oak, fue concebido como un lenguaje para desarrollo de aplicaciones para aparatos de casa y otros dispositivos. Con la revolución de Internet, Java gradualmente se envolvió en un lenguaje desarrollado para lado del cliente con capacidades como los Applets y JavaBeans. Con el tiempo, muchas API's, como el Java Database Connectivity (JDBC), fué desarrollado para direccionar necesidades de mercado para acceso genérico y el uso de recursos típicamente requeridos por aplicaciones de software empresarial.

Esto fue claro después de que Java introdujera el uso de ambientes de sistemas basados en browser en el lado del cliente, donde encaró algunos retos serios, como la tardanza que envuelve

el hecho de cargas la librería java a través de Internet antes de que la aplicación java pueda iniciar. Sin embargo, la relativa sencillez de Java, la arquitectura de plataforma independiente, y el buen conjunto en API's también como su naturaleza Web donde es fuertemente positivo su uso dentro del desarrollo de software empresarial.

Esta comodidad del uso y la Web permitió que Java se adoptara relativamente para desarrollos Web-céntrico. Los desarrolladores usan la tecnología Java, como Applets, para salidas dinámicas y visuales que pudieran hacer fácil, el agregarlo en páginas HTML sobre sitios Web.

Pero también las aplicaciones Java corren sobre servidores, Java inicialmente no ofrecía ninguna capacidad específica de uso del lado del servidor. Sun realizó un esfuerzo para usar Java como lenguajes para aplicaciones basadas en Web y procuró adaptar este lenguaje para trabajar del lado del servidor vía las especificaciones Java Servlet. Una vez que la adaptación ocurrió, el cliente Web podría llamar a un programa Java corriendo sobre un servidor remoto, y el programa servidor podría procesar la petición y regresar el resultado obtenido de la petición. El concepto "El Servlet" hació y ha sido utilizado fuertemente para el desarrollo de aplicaciones empresariales. Los Servlets, sin embargo, nunca fueron diseñados para manejar realmente los asuntos complejos relacionados a transacciones de cliente, concurrencia de sesiones, sincronización con datos, etc.

EJB, originalmente lanzado como una especificación independiente por Sun Microsystems, fue empleado para simplificar desarrollo del lado del servidor proveyendo un gran numero de conjuntos de servicios "fuera de la caja" para manejar características desarrolladas para aplicaciones empresariales.

El concepto de la arquitectura en capas ha estado dando vueltas por mucho tiempo, y ha sido usada para construir aplicaciones a escala empresarial. Sun reforzó mucho la idea de desarrollar el modelo de una arquitectura en n-capas con Java, e introdujo funcionalidad específica para permitir el fácil desarrollo del lado del servidor de aplicaciones empresariales escalables basadas en Web, dándole a Java el ingrediente crítico necesario para esta arena.

El J2EE es el resultado de esfuerzos de Sun para alinear las separadas tecnologías Java y API's en una cohesiva plataforma de desarrollo Java para desarrollar tipos específicos de aplicaciones. Actualmente existen 3 plataformas Java existentes.

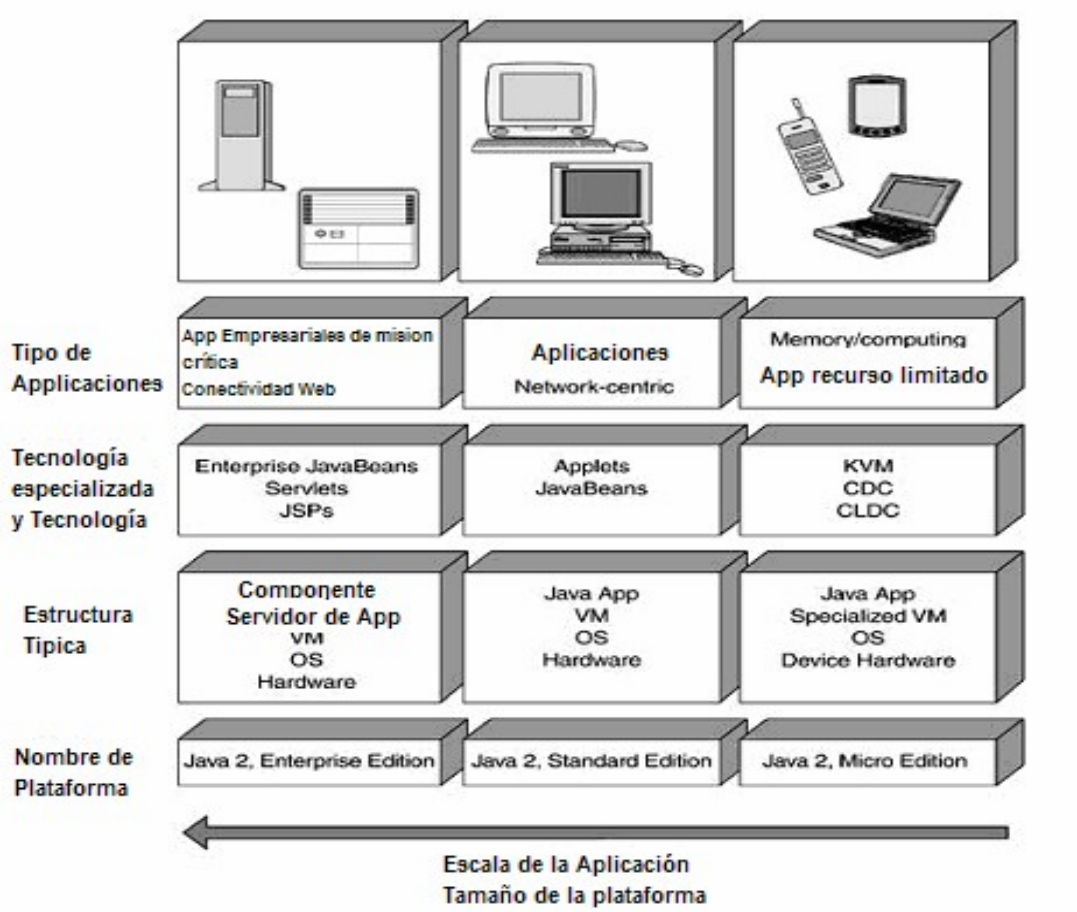

Figura F2.14 Modulos relacionados con el J2EE, J2SE y J2ME

2.9.1 Porque J2EE?

Probablemente uno se pregunta, pero ¿Porque usamos J2EE?, ¿Que no es muy nuevo y todavía no probado?, ¿Que realmente ofrece?....

Comencemos por los aspectos generales del J2EE, por ejemplo, el API JDBC está bien establecido, y es una característica indispensable para conexión con Base de Datos. La tecnología Servlet ha sido incluso usada por algún tiempo como una ligera alternativa para los Scripts Common Gateway Interface (CGI), estos Scripts son como un acercamiento viejo para procesar entradas de información proveídas por el usuario vía la Web y para proveer contenido dinámico basado en la entrada.

J2EE ofrece incluso algunos beneficios prometedores, incluye características para permitir a los desarrolladores a enfocarse en desarrollar lógica de negocio, sobre la implementación del sistema sin antes tener conocimiento detallado del ambiente de ejecución. Y sobre la creación de sistemas que son portables entre plataformas de hardware y sistemas operativos diferentes.

Desarrollar software empresarial es una tarea compleja y requiere grandes conocimientos de muchas áreas diferentes. Por ejemplo, un típico desarrollo de aplicación empresarial requiere que uno esté familiarizado con aspectos de de comunicación entre procesos, aspectos de seguridad, consultas especificas de acceso a base de datos, etc. El J2EE incluye un transparente e integrado soporte para este tipo de servicios. Como resultado, desarrolladores están dispuestos a enfocarse sobre implementación lógica de negocio más que código que soporte infraestructura de aplicación básica.

El modelo de desarrollo empresarial J2EE tiene una división más limpia entre el desarrollo del sistema, desplegado, y ejecución. A causa de esto, los desarrolladores definen detalles de despliegue, tal como el nombre verdadero de la base de datos y la ubicación, reciben las propiedades específicas de configuración, etc., hacia el desplegador.

J2EE soporta independencia de hardware y Sistema Operativo(OS) habilitando servicios de sistema para ser accesado vía Java y por API's de sistemas J2EE. Por esta razón, los sistemas empresariales que conforman la especificación de arquitectura J2EE es portada muy fácil entre diferentes sistemas de hardware y diferentes sistemas operativos. Quizás uno de los grandes beneficios del J2EE es su uso para componentes.

El software basado en componentes tiene numerosas ventajas sobre el tradicional desarrollo de sistemas:

- Alta Producción: Pocos desarrolladores logran más poniendo una aplicación pre-construida, componentes probados con anterioridad antes que aplicando una solución personalizada que choque.
- Desarrollo Rápido: Componentes existentes son puestos juntos rápidamente para crear nuevas aplicaciones.
- Alta Calidad: Antes que probar aplicaciones enteras, los desarrolladores de aplicaciones basados en componentes se concentran en probar la integración y toda la funcionalidad de la aplicación vía componentes pre-construidos.
- Mantenimiento Fácil: Como los componentes son puestos solos (autónomos) para empezar, mantenimiento como actualizaciones de componentes individuales es mucho mas fácil y más rentable.

J2EE facilita el basarse en componentes en muchos sentidos, como lo siguiente:

- La naturaleza "escribe una vez, y corre en donde sea "de Java, lo hace atractivo para los desarrolladores de componentes por la compatibilidad con un diverso conjunto de sistemas de hardware y sistemas operativos.
- J2EE ofrece un buen enfoque para separar los aspectos de desarrollo de componentes de su especificación de ensamble y sus aspectos de ensamble. Esto es, los componentes desarrollados independientemente se integran prontamente en ambientes y aplicaciones nuevos.
- J2EE ofrece un amplio rango de API's a ser usados para acceder e integrar productos, por ejemplo, base de datos, sistemas de correos, plataformas de mensajes, etc.

- J2EE ofrece componentes especializados que son optimizados para tipos específicos de roles en una aplicación empresarial. Por ejemplo, componentes empresariales son desarrollados de diferentes "sabores," dependiendo de que es lo que ellos quieren completar.

2.10 Tecnología J2EE

2.10.1 Contenedor

Para entender la tecnología J2EE, es necesario entender en primera instancia el rol de los contenedores dentro de la arquitectura. Todas las tecnologías actuales en el J2EE dependen de este concepto sencillo pero poderoso. (ver Figura F2.15)

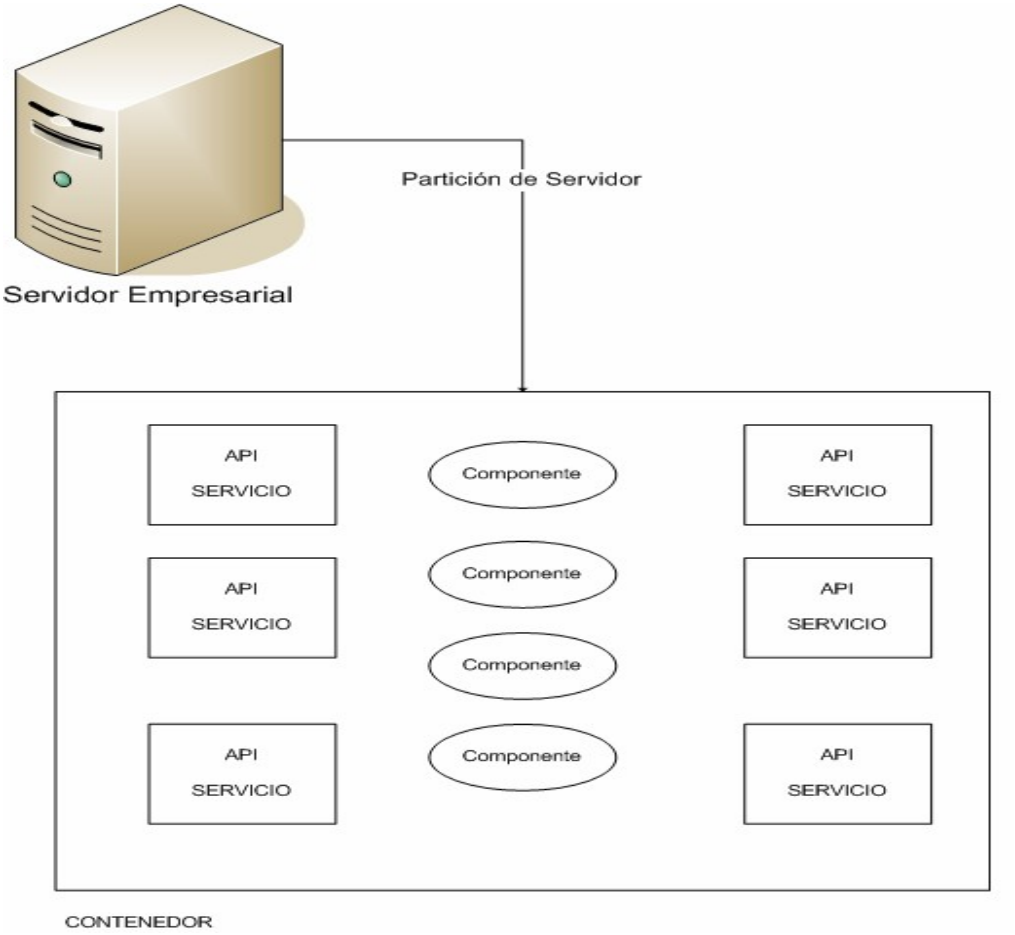

Figura F2.15 Gráfica del concepto de contener

Un contenedor es una entidad de software que corre dentro de un servidor y es responsable de manejar tipos específicos de componentes. Este provee un ambiente de ejecución para componentes J2EE que se desarrollen. Esos contenedores proveen independencia entre desarrollo y despliegue y proveen portabilidad entre diversas capas del servidor.

Un contenedor incluso es responsable de manejar los ciclos de vida de los componentes desplegados dentro del y cosas como el almacenamiento de recursos y reforzado de seguridad. Por ejemplo, se restringe la habilidad de acceder un método en específico para un grupo pequeño de solicitantes. El contenedor refuerza esta restricción interceptando solicitudes para un método en específico y asegurando que esa entidad que está tratando de acceder está en la lista de privilegiados.

Dependiendo del tipo de contenedor, proveerá acceso a algunos o todos los API's J2EE. Todos los componentes J2EE son desplegados y ejecutados dentro de algún contenedor. J2EE tiene 4 tipos diferentes de contenedores:

- Contenedor de Aplicaciones: Hospeda aplicaciones Java independientes
- Contenedor Applet: Provee un ambiente de ejecución para Applets
- Contenedor Web: Almacena los componentes Web, como Servlets y JavaServer Pages (JSP)
- Contenedor Empresarial: Almacena componentes EJB

Los contenedores proveen soporte en tiempo de ejecución para componentes de aplicación J2EE; también proveen una visión mas dividida de los ocultos API's J2EE a los componentes de aplicación..

Las herramientas del contenedor deben entender los formatos de archivos para el empaquetado de componentes de aplicación para su despliegue.

2.10.2 Servlets

Los Servlets son componentes Web capaces de generar contenido dinámico. Son uno de los componentes J2EE más usados en el World Wide Web hoy en día y proveen un mecanismo efectivo para interacción entre la lógica de negocio basada en servidor y el cliente basado en Web, y proveen una ligera y mas manejable alternativa a los populares Scripts CGI's.

Porque los Servlets son simples y requieren pocos recursos en general, algunos desarrolladores prefieren usar estos componentes en conjunto con los JSP casi exclusivamente en sus implementaciones antes de utilizar los componentes EJB. Esta práctica quizás tenga sentido para aplicaciones muy sencillas de empresa, pero rápidamente llegó a ser una opción más o menos óptimo cuando el soporte transaccional es necesario en una aplicación.

Los Servlets son por lo general usados para manipular simples tareas, como el recibir y verificar entradas validadas de los campos de entrada de una página Web. Cuando los chequeos preliminares se hacen, los datos deben ser pasados a un componente mucho más adecuado para

realizar tareas actuales a mano, estos corren dentro de un contenedor (incluso es llamado un motor Servlet) hospedado sobre un servidor Web. El contenedor Servlet maneja el ciclo de vida de un Servlet y traduce las peticiones Web desde el cliente, hecho vía protocolos como el Hypertext Transfer Protocol (HTTP), dentro de peticiones basadas en objetos. Igualmente, el contenedor traduce la respuesta de un Servlet y envía la respuesta al protocolo de Web apropiado.

2.10.3 JSP

Los JSP son otro tipo de componente Web y ha envuelto a la tecnología Servlet. De hecho, partes del JSP son compilados en Servlets que son ejecutados dentro de un ambiente contenedor Servlet.

JSP's vinieron a ser mas fáciles para los miembros del equipo Web, para mantener las porciones del sistema que soporta páginas Web, sin tener precisamente que ser un programador tradicional. Un no programador típicamente mantiene el código de presentación en HyperText Markup Language(HTML). Esto es difícil de hacer cuando el HTML es generado en Java dentro de un Servlet.

Los JSP permiten codificar en lenguaje Java y estos códigos ser embebidos dentro de una estructura como HTML, o como eXtensible Markup Language (XML). Esto permite al código de presentación ser más fácil de mantener como un HTML regular y protegiendo contribuciones no técnicas desde editores de código.

Debido a que los JSP's permiten código Java muy complejo para ser embebido dentro de documentos HTML Y XML, algunos desarrolladores escogen usar este método de desarrollo de sistemas. Es generalmente buena práctica el mantener el código java dentro de un JSP relativamente simple.

2.10.4 EJB

La especificación EJB está muy en el centro de la plataforma J2EE. Define un modelo completo y comprensible para construir componentes de aplicación Java basadas en servidor de forma distribuida y escalable.

Existen 3 tipos de EJB's:

- Los Session beans son los más usados para actividades transitorias. Son no persistentes y ofrecen encapsular la mayor parte de la lógica de negocio dentro de una aplicación empresarial. Estos EJB's son "stateful," significando que ellos retienen conexiones entre interacciones sucesivas con el cliente. El otro tipo de EJB son "stateless," donde cada invocación del session bean por el mismo cliente es tratado como nuevo.
- Los Entity beans encapsulan datos persistentes en un data store, que es típicamente un registro parcial o total de información encontrado en la base de datos. Los Entity beans proveen servicios automatizados para asegurar que la vista orientada a objetos de esos

datos persistentes se mantenga sincronizado en todo momento con el actual residente de datos en la base de datos oculta. Estos EJB's incluso son usados para formatear datos, ya sea para asistir en la lógica de negocios, tarea a mano o para preparar los datos para mostrarlos en una página Web, por ejemplo, en una base de datos suponiendo que se tiene una tabla llamada empleados, cada registro puede mapearse a una instancia de un Entity bean.

- Los Message-driven beans se diseñan para ser consumidores convenientes y asíncronos de mensajes de servicio (JMS) de mensajería de java. Los message-driven beans no tienen interfaces Públicadas, si no que los message-driven beans operan anónimamente detrás de las escenas, también son stateless (cada invocación es tratada como nueva).

Existe una arquitectura de tipo Model-View-Controller (MVC), originalmente usada en el lenguaje de programación Smalltalk, que es muy útil para ejemplificar la tecnología J2EE. Básicamente la idea del MVC es balancear funciones, alineándolo con sus respectivos conjuntos de responsabilidades en el área de datos persistentes y reglas asociadas (Modelo), presentación (Vista), y la lógica de aplicación (Controlador) (ver figura F2.16).

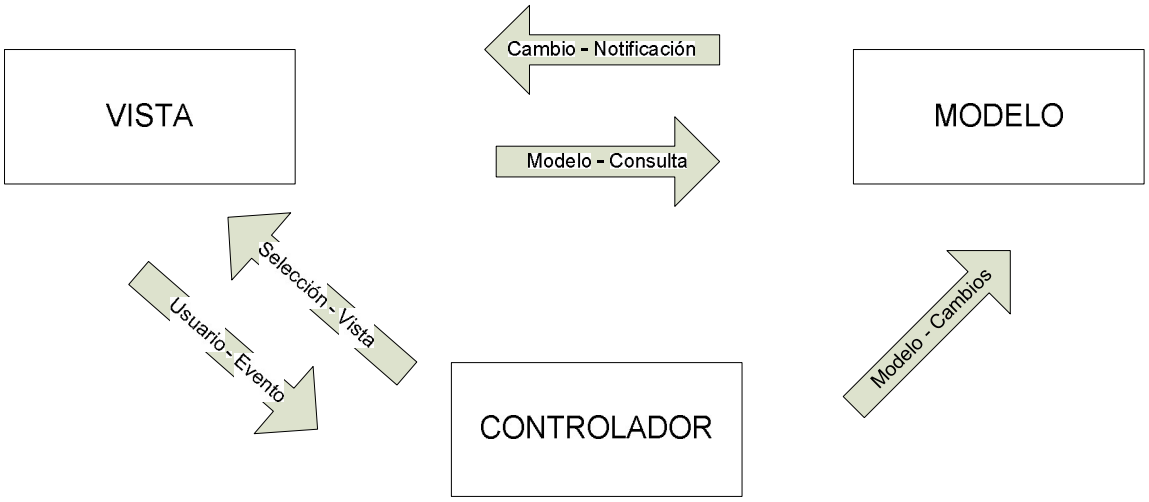

Figura F2.16 Esquema Vista –Modelo - Controlador

El modelo es responsable de mantener el estado de la aplicación y datos. Éste puede recibir y responder a consultas desde la Vista y provee notificaciones a la vista cuando las cosas han cambiado.

El controlador actualiza el modelo basado en la aplicación de la lógica de negocios en respuesta a las acciones del usuario (ejemplo diálogos, peticiones de formularios, etc). Éste es también responsable de decirle a la vista que desplegar en respuesta a acciones del usuario.

La Vista es responsable de la interpretación verdadera de los datos proporcionados por el Controlador.

Para hacer más explicativa esta parte, consideremos una simple aplicación de reloj usando MVC. El Modelo en estos casos es esencialmente responsable de mantener el seguimiento del tiempo, el tiempo es automáticamente actualizado en intervalos predefinidos (un microsegundo, milisegundo, o alguna otra cantidad) por algunos mecanismos incorporados en el Modelo. Esto incluso provee operaciones que otras entidades pueden consultar el modelo y obtener el tiempo actual, pero no importa o no se sabe como el tiempo va a ser desplegado.

La responsabilidad para desplegar el tiempo cae en la vista; la vista toma varias formas. Por ejemplo, puede tomar la forma de un reloj análogo, desplegando el tiempo por medio de 2 ó 3 manecillas. Este también es desplegado por muchos dígitos. Conforme el tiempo cambie, el modelo notifica a la vista, y la vista actualiza y refleja el nuevo tiempo.

Debemos de tener bien en mente que el reloj requiere de algún mecanismo para actualizar el tiempo, por ejemplo, cuando el horario de verano entra o sale. En un reloj reflejado en un navegador Web, el usuario tiene la capacidad de indicar un cambio en el tiempo usando alguna interfaz de usuario (Graphical User Interface GUI), ya sea controlando eso, o cambiando directamente la hora. El control recibe esas acciones del usuario y actualiza el Modelo llamando a la operación adecuada definida sobre el modelo, para reflejar el nuevo tiempo.

2.10.5 API's

Existen muchos API's dentro del J2EE. Los más comunes son descritos a continuación:

2.10.6 JDBC

La interacción con una o mas bases de datos es una parte integral de una aplicación java. El API JDBC esta prácticamente enfocado en hacer fácil la interacción de la base de datos para el desarrollador Java.

El API JDBC, que es muy similar en espíritu al Microsoft Open Database Connectvity (ODBC), simplifica el acceso a la base de datos relacional. Usando el JDBC hace a una aplicación portable y con las habilidades de la base de datos aplicables.

En aplicaciones empresariales, no necesariamente se necesita usar el JDBC directamente, por ejemplo, se puede usar un Entity bean para hacer las llamadas necesarias a la base de datos.

2.10.7 Java Naming and Directory Interface (JNDI)

Servicio de denominación (naming) permite poder consultar (referir a) un objeto. Un sistema de archivos es un ejemplo de un servicio de denominación.

Un servicio de directorio es muy similar a un servicio de denominación y provee mejora en capacidades de búsqueda. De hecho, un servicio de directorio siempre tiene un servicio de denominación (pero no viceversa).

2.10.8 Java Message Service (JMS)

Un servicio de mensajeria permite comunicar entre aplicaciones distribuidas usando entidades auto contenedoras llamadas mensajes. Dada comunicación es típicamente asíncrona. El JMS provee una uniforme y genérica interfaz para dado middleware(capas de lógica de negocios, o componentes Web).

EL JMS es utilizando directamente en una aplicación empresarial o via un tipo de EJB (message-driven).

2.10.9 Remote Method Invocation (RMI)

RMI habilita el acceso a componentes en un ambiente distribuido, permitiendo a los objetos java invocar métodos sobre objetos Java remotos. El método se invoca realmente sobre un objeto Proxy, que entonces arregla para pasar el método y parámetros a un objeto remoto y provee la respuesta desde el objeto remoto hacia el objeto que ha iniciado la invocación del método remoto.

2.10.10 Otras tecnologías J2EE y API´s

Aquí se listarán algunas de las otras tecnologías J2EE y API´s que están ya sea en existencia actualmente o se espera que sea parte del J2EE en un futuro.

2.10.11 Conectores J2EE

Los conectores J2EE proveen una arquitectura en común a usar cuando se trate con sistemas de información empresarial (EIS) como almacén de datos. Estos sistemas grandes tienden a ser predominantes en empresas grandes, y es muy complejo tratar con ellos.

2.10.12 Java Transaction API (JTA)

El JTA provee un API genérico y de alto nivel para el manejo de transacciones. Esto es primordialmente usádo procesamiento de transacciones grandes, complejas y distribuidas, usualmente involucrando un gran número de conexiones de sistemas remotas. Una transacción se refiere a múltiples operaciones en una simple tarea, esto es, si parte de la transacción falla, las otra previa operación ejecutada es puesta en marcha, esto es deshacer, para asegurar la sanidad del sistema.

2.10.13 Java IDL

La Java Interface Definition Language (IDL) provee soporte de interoperabilidad para la Common Object Request Broker Architecture (CORBA) y para la industria estándar Internet Inter-Orb Protocol (IIOP). Esto incluye un compilador IDL-to- Java y un ligero Object Request Broker(ORB).

2.10.14 RMI-IIOP

El RMI-IIOP se refiere a RMI usando el IIOP como un protocolo de comunicación bajo cubierto. IIOP es un entandar Object Management Group (OMG). Porque CORBA utiliza IIOP como protocolo fundamental, el uso de RMI-IIOP hace interoperabilidad entre el RMI y CORBA.

2.10.15 Java Transaction Service (JTS)

JTS es un servicio manejador de transacciones que soporta JTA y hace uso de IIOP para comunicarse entre instancias remotas del servicio. Igual que JTA, este es usado en situaciones con grandes sistemas distribuidos.

2.10.16 JavaMail

JavaMail provee un API para facilitar interacción con los sistemas de mensajes e-mail. Este API consiste primordialmente de un conjunto de clases abstractas que modela un sistema de e-mail basado en java. Esta hecho para construir aplicaciones sofisticadas basada en e-mail. Pero hay que tener en claro que es posible proveer soporte para e-mail en una aplicación sin la necesidad forzosa de usar el API JavaMail.

2.10.17 Servicios de Seguridad

El Java Authentication and Authorization Service (JAAS) habilita servicios para autenticar y para reforzar los controles de acceso a usuarios. Esto implementa una versión de tecnología Java del estándar Plegable Authentication Module (PAM), y entiende la arquitectura de control de acceso de la plataforma Java 2 de una forma compatible. El Java Authorization Service Provider Contract for Containers(JACC) define un contrato entre un servidor de aplicaciones J2EE y un proveedor de servicios de autorización , permitiendo proveedores de servicios de autorización personalizables para ser conectado en un producto J2EE.

2.10.18 Web Service

El J2EE provee total soporte para ambos clientes del Web Services. Varias tecnologías Java trabajan en conjunto para proveer soporte para Web Service. EL API para XML basado en RPC

(JAX-RPC) provee soporte para llamados a Web service usando el protocolo SOAP/HTTP. JAX-RPC define el mapeo entre clases Java y XML para ser usado como llamada SOAP RPC. EL SOAP con el Attachments API for Java(SAAJ) provee soporte para manipular mensajes SOAP de bajo nivel. Los Web Services para la especificación J2EE define totalmente el despliegue de clientes Web Service y puntos finales Web Service en J2EE, también la implementación de Web Service usando Enterprise beans. El Java API for XML Registries (JAXR) provee acceso cliente a servidores XML registrados.

2.10.19 Relación de Elementos de Arquitectura J2EE

La relación de los elementos de arquitectura de la plataforma J2EE, son mostrados en la figura F2.17. La figura muestra las relaciones lógicas de los elementos, esto no significa que implique un particionamiento físico de los elementos en máquinas separadas, procesos, espacios direccionados, o máquinas virtuales.

Los contenedores, denotados por los rectángulos separados, son ambientes en tiempo de ejecución J2EE que proveen los servicios necesarios para los componentes de aplicación representados en la mitad de los rectángulos mostrados en la parte superior. Los servicios proveídos son denotados por las cajas en la parte baja, de la misma manera en la mitad del rectángulo.

Las flechas representan el acceso requerido a otras partes de la plataforma J2EE. El contenedor de aplicación cliente provee Aplicaciones cliente con acceso directo a base de datos J2EE requeridas a través del API de Java para conectividad con los sistemas de base de datos, el API JDBC. Acceso similar a la base de datos es proveída por la páginas JSP y Servlets por el contenedor Web, y para los Enterprise beans por el contenedor de EJB.

Como indican los API's de la plataforma Java 2, Standard Edition (J2SE), son soportados por los ambientes runtime J2SE para cada tipo de componentes de aplicaciones

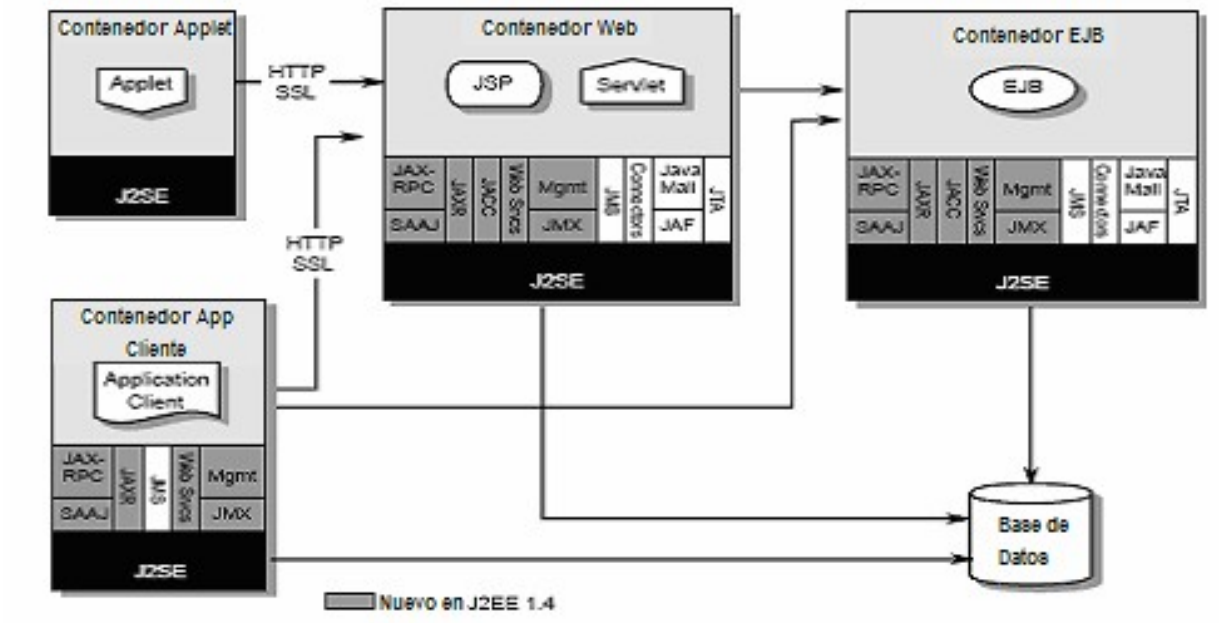

Figura F2.17 Elementos de una arquitectura J2EE

A continuación se describirán los requerimientos de la plataforma J2EE para cada elemento:

2.10.20 Componentes de aplicación

El ambiente J2EE define 4 componentes de aplicación que soporta:

- Las aplicaciones cliente son programas hechos en lenguaje Java que son típicamente programas GUI que se ejecutan en una computadora de escritorio. Las aplicaciones cliente ofrecen una experiencia similar al que ofrecen las aplicaciones nativas, y tienen acceso a todas las facilidades de la capa media del J2EE.
- Los Applets son componentes GUI que son típicamente ejecutados en un browser, pero ejecutan una variedad de aplicaciones o dispositivos que soporten el modelo de programación del Applet. Los Applets son usados para proveer una interfaz de usuario poderosa para aplicaciones J2EE. (las simples páginas HTML son usadas para proveer una interfaz de usuario más limitada para aplicaciones J2EE)
- Los Servlets, páginas JSP, filtros, y los eventos Web listeners típicamente se ejecutan en un contenedor Web y responden a peticiones HTTP desde clientes Web. Servlets, páginas JSP, y filtros son usados para generar paginas HTML que son interfaces de aplicaciones. Esos son incluso usados para generar XML u otro formato de datos que son usados por otros componentes de aplicación. Un tipo especial de Servlet provee soporte para Web Services usando protocolo SOAP/HTTP. Los Servlets, páginas creadas con tecnología JavaServer Pages, filtros Web, y los Web event listeners todos son referenciados colectivamente en la especificación J2EE como "Componentes Web (Web components)." Las aplicaciones Web tienen componentes Web y otros datos como las paginas HTML. Los componentes Web se ejecutan en un contenedor Web. Un servidor

Web incluye un contenedor Web y soporta otros protocolos, soporta seguridad, y es un requerimiento de la especificación J2EE.

- Los componentes Enterprise JavaBeans (EJB) se ejecutan en un ambiente manejado que soporta transacciones. Los Enterprise beans típicamente contienen lógica de negocio para aplicaciones J2EE. Los Enterprise beans proveen directamente Web Services usando protocolo SOAP/HTTP.

2.10.21 Rol de funciones en una plataforma J2EE

2.10.21.1 Proveedores de componentes de Aplicación

Dentro del tema de los proveedores, siguiendo lo planteado en la especificación J2EE, existen muchos roles para la parte de componentes de aplicación, entre ellos están los diseñadores de páginas HTML, programador de documentos, desarrolladores de Enterprise beans, estos roles usan herramientas para producir aplicaciones y componentes J2EE.

2.10.21.2 Ensamblador de Aplicaciones

El ensamblador de aplicaciones toma un conjunto de componentes desarrollados por los proveedores de componentes de aplicaciones, los reúne dentro de una completa aplicación J2EE en forma de un archivo empresarial (.ear). El ensamblador de aplicaciones generalmente usa herramientas GUI proveídas por un proveedor de plataforma o por un proveedor de herramientas. El ensamblador de aplicaciones es responsable de proveer instrucciones en forma describiendo dependencias externas de las aplicaciones que el desplegador resuelve en el proceso de desplegado.

2.10.21.3 Desplegador (Deployer)

Es responsable de desarrollo de desplegar aplicaciones cliente, aplicaciones Web y componentes Enterprise JavaBeans dentro de un ambiente operacional específico. El desplegador usa herramientas suplidas por algún proveedor de productos J2EE para realizar ese tipo de tareas. El despliegue es típicamente un proceso de 3 etapas:

1. Durante la instalación, el desplegador mueve aplicaciones multimedia al servidor, genera clases especificas del contenedor adicionales e interfaces para habilitar el contenedor para manejar los componentes de aplicación a la hora de correrlos, también instala componentes de aplicación, junto con clases adicionales e interfaces, dentro de los contenedores J2EE correctos.
2. Durante la configuración, dependencias externas declaradas por el proveedor de componentes de aplicación son resueltas e instrucciones reunidas de aplicación definidas por el ensamblador de aplicaciones son seguidas. Por ejemplo, el desplegador es responsable de mapear los roles de seguridad definidos por el ensamblador de

aplicaciones dentro de grupos de usuarios y cuentas que existen en un ambiente operacional.
3. Finalmente, el desplegador empieza una ejecución de la aplicación instalada y configurada.

En algunos casos, un desplegador calificado configura la lógica de negocio de algunos componentes de aplicación en el momento en que se haga el despliegue. Por ejemplo, en desplegador proveído de herramientas necesarias hace un código para agregar uno o varios métodos de negocio en los Enterprise beans, o personalizar la apariencia de una página JSP.

En realidad lo que un desplegar puede hacer o modificar, son aplicaciones Web, Enterprise beans, y aplicaciones cliente que han sido personalizadas para el ambiente operacional y son puestos en un contenedor J2EE específico.

2.10.21.4 Administrador del sistema

El administrador del sistema es responsable de la configuración y administración de la infraestructura computacional y de red empresarial. El administrador del sistema es incluso responsable de revisar el estatus de las aplicaciones, también típicamente usa herramientas de manejo y monitores en tiempo real para completar sus tareas.

2.10.21.5 Proveedor de Componentes de Sistema

Una variedad de componentes de sistema son facilitados por los proveedores de componentes, la arquitectura de conectores definen los APIS's primarios usados para proveer adaptadores de recursos de muchos tipos. Estos adaptadores de recursos se conectan a sistemas de información empresarial de muchos tipos, incluyendo base de datos y sistemas de mensajería.

2.10.22 Tecnología J2EE en resumen.

J2EE ofrece una arquitectura bien pensada para desarrollar aplicaciones complejas de la empresa usando Java. La combinación de tecnologías J2EE (llamadas EJB, Servlets, JSP's, y sus API's genéricos), le da a sus usuarios varias ventajas. Entonces, desarrollando una aplicación J2EE simplifica la tarea general de desarrollar aplicaciones distribuidas a grande escala.

Algunos de los retos claves que son implementados por el J2EE incluyen la distribución de aplicaciones a través de múltiples procesos y procesadores, seguridad, transacciones, manejo de persistencia, y despliegue.

2.11 ¿Porqué usar el J2EE y el UML juntos?

Cualquier programador desarrolla una pieza de software que hace el trabajo por un tiempo; la construcción de un sistema empresarial que es mantenible, escalable, y evolvente son asuntos enteramente diferentes. Cuando un sistema entra a una etapa crítica, es decir, que se requiera ampliar urgentemente su funcionalidad, que nuevos módulos requieran ser agregados, existiera retrasos en la construcción de algún módulo, etc; es muy importante visualizar todos los componentes, porque se necesitara mantener, escalar y envolver el sistema que se está construyendo.

Es posible sobrevivir y prosperar por un rato codificando, compilando, arreglando y desplegando la aplicación, tarde o temprano, esa aplicación mostrará que no es escalable o los nuevos niveles de demanda. Esto se debe a que posiblemente no tiene la arquitectura y diseño que podría hacer mas fácil la escalabilidad de los nuevos requerimientos.

El UML provee las herramientas necesarias para diseñar y construir sistemas complejos. Sostiene, entre otras disciplinas, los requisitos que dirigen, el diseño de la arquitectura-nivel, y el diseño detallado. Las herramientas de modelado UML están evolucionando donde son usadas para imponer patrones consistentes de diseño sobre modelado de sistemas basado en J2EE y para generar una parte considerable de código fuente del sistema.

Usando UML para modelado de requerimientos, en conjunción con un caso de uso conducido por un proceso de desarrollo, facilita el trazar requerimientos hacia el diseño. El trazar en este contexto, implica la habilidad para determinar los elementos en un diseño que existe como un resultado de un requerimiento en específico. Un caso de uso conduce procesos de desarrollo, elementos de diseño específico son creados para el propósito de satisfacer el caso de uso.

El medir requerimientos tiene varios beneficios. Por ejemplo, la habilidad para identificar el impacto de un cambio de requerimientos sobre el diseño, esto es no solo simplificar la tarea de modificar un sistema para conocer los nuevos requerimientos, si no ayudar a enfocar pruebas del sistema después de que los cambios están completados.

UML incluye constructores de modelado que ayudan a los desarrolladores a entender como grandes pedazos de sistema interactúan en tiempo de ejecución y dependen unos de los otros en tiempo de compilación. Las herramientas de UML incluyen chequeos para asegurar que los detalles del diseño no violan los límites de la arquitectura.

Los diagramas UML, como los diagramas de interacción, diagramas de actividad, y diagramas de clase, son usados para entender y documentar interacciones complejas dentro del sistema. Esto ayuda en el análisis del problema e incluso provee un record detallado de la conducta y estructura del sistema. Cuando es tiempo de incorporar nuevas funcionalidades en el sistema, sabemos cual era la intención de diseño y que limitaciones de sistema hay.

Finalmente, usando UML permite a los desarrolladores moverse hacia un paradigma de desarrollo visual. En adición, permitirá los desarrolladores imponer un patrón de modelo consistente en sus diseños, las herramientas de modelado moderno UML genera un incremento en la alta

funcionalidad del código fuente J2EE. Como resultado, los desarrolladores se concentran sobre actividades de diseño de alto valor y dejar mucho trabajo de la codificación a las herramientas de modelado. Una representación es incluso excelente para comunicar el diseño entre equipos.

Capítulo 3

3 Propuesta de una arquitectura de sistemas empresarial en capas totalmente basada en software libre

3.1 Arquitectura en capas

Siguiendo lo visto en el capitulo 2, podemos afirmar que se conoce ya la arquitectura de una aplicación empresarial, pero ¿Qué productos basados en software libre se usan para cumplir los estándares vistos, que cubra todas y cada una de las necesidades que la empresa necesita?, para resolver esta pregunta, tenemos primero que partir de la idea de que necesitamos que nuestra arquitectura esté basada en capas.

Se nombrara cada software, con su respectivo módulo con el cual trabaja, dentro de la arquitectura en capas, para nuestro ejemplo nos basaremos en una arquitectura que se denominará de 4 capas(Web browser cliente, Servidor Web y parte del Servidor de Aplicaciones, Lógica de Negocios del Servidor de Aplicaciones y Datos) como se muestra en la figura F3.1.

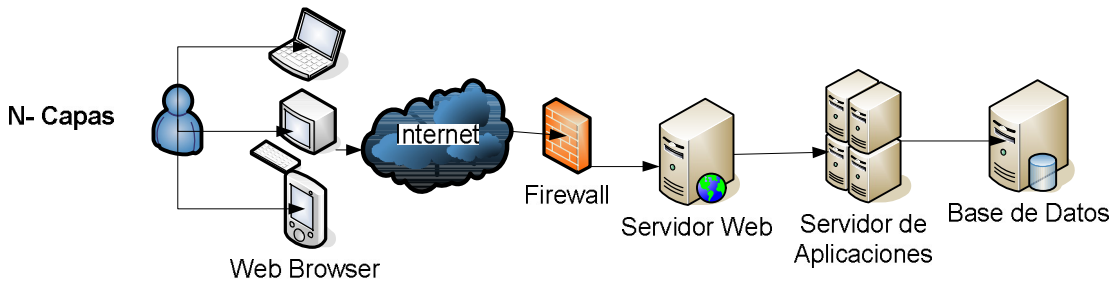

Figura F3.1 Arquitectura con n - capas

Las características que tiene una aplicación de 4 capas es el poder separar los datos, la lógica de negocio, la presentación y el cliente (la diferencia entre presentación y cliente es de que presentacion contiene los archivos html encargados de presentar lo que el cliente esta solicitando, y el cliente, en el caso de las aplicaciones web, esta conformado por el browser, encargado de desplegar lo que el servidor web le envie) en para fines de orden, administración y seguridad. Al tener los datos en una capa distinta a la de presentación permite no estar en contacto directamente con ningún usuario lo que implica mayor seguridad, la forma de acceder sería por medio de la capa de lógica, la cual esta previamente limitada a extraer ciertos datos de interés. La ventaja de tener la capa de presentación separada de la lógica de negocio, permite evitar cualquier tipo de acceso no autorizado al funcionamiento de la aplicación. La capa de

presentación del usuario, permite el poder acceder a la aplicación vía Web sin necesidad de tener algún tipo de aplicación cliente, es decir, toda petición que realice el usuario será atendida y enviada por el servidor Web al servidor de Aplicaciones.

El servidor de aplicaciones se compone de los módulos ilustrados en la figura F3.2:

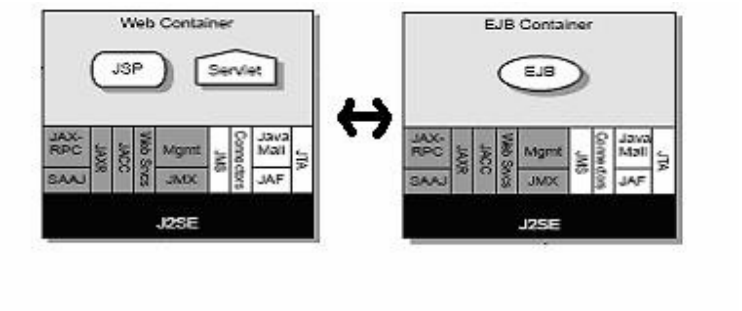

Figura F3.2 Modulos que conforman un servidor de aplicaciones

El contenedor Web es parte, para nuestro ejemplo, junto con el servidor Web de la capa de presentación, donde ésta se comunica con la capa de lógica de negocio y ésta a su vez con la capa de almacén de datos.

Conceptualmente hablando se definen las capas como muestra la figura F3.3 (sin tomar en cuenta el aspecto físico), arquitecturas similares son usadas comúnmente hoy en día en muchas empresas mexicanas.

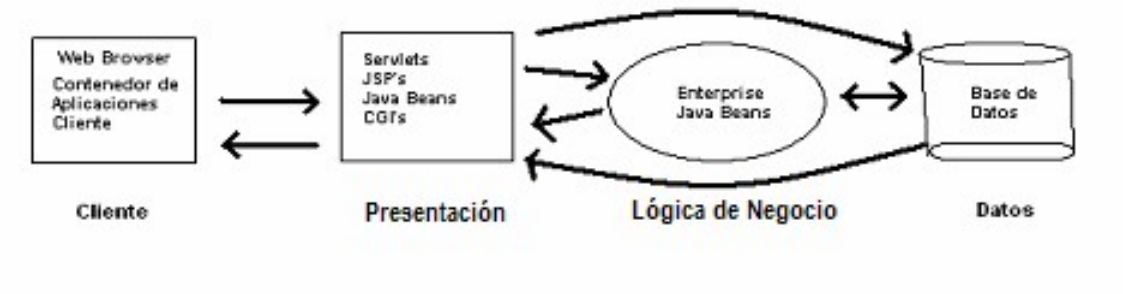

Figura F3.3 Detalle de arquitectura con 4-capas

También hay que agregar la plataforma de sistema operativo que es vital para montar todo este concepto, el sistema operativo no es una capa, pero es vital ya que funge como plataforma sobre la cual trabajan las capas.

3.1.1 Sistema Operativo Linux

Todos los administradores de ventanas importantes han sido incorporados a Linux. Todas las utilidades de Internet, como el protocolo de transferencia de archivos FTP, los navegadores de Web, y las conexiones remotas con PPP, son parte integrante de Linux. Cuenta, además, con una completa serie de utilidades de desarrollo de programas, como los depuradores, y compiladores C++. Debido a todas sus características, el sistema operativo Linux sigue siendo pequeño, estable y rápido. En su formato más simple, Linux ejecuta efectivamente en tan sólo 4 mb de memoria.

A pesar de que Linux ha evolucionado en el entorno abierto y gratuito de Internet, se apega a los estándares oficiales de UNIX. Debido a la proliferación de versiones UNIX durante las décadas anteriores, el IEEE (Institute of Electrical and Electronics Engineers) desarrolló un estándar independiente de UNIX para el ANSI (American Nacional Standards Institute), se denominó a este nuevo estándar de UNIX POSIX. El estándar define como debe de operar un sistema de tipo UNIX, para lo que especifica detalles como las llamadas del sistema y las interfaces. POSIX define un estándar universal con las que deben cumplir todas las versiones de UNIX. Linux fue desarrollado desde un principio con el estándar POSIX, y fue concebido por medio de estándares abiertos.

A diferencia de UNIX, Linux tiene la capacidad de dividirse, generalmente, en tres componentes principales: el Kernel, el entorno y la estructura de archivos. El Kernel es el programa base que ejecuta programas y administra los dispositivos hardware, como por ejemplo, los discos y las impresoras. El entorno proporciona una interfaz para el usuario; recibe comandos del usuario y los envía al Kernel para que sean ejecutados. La estructura de archivos organiza el modo en que se almacenan los archivos en un sistema de almacenamiento, por ejemplo, un disco. Juntos, el Kernel, el entorno y la estructura de archivos forman la estructura básica del sistema operativo.

Como alternativa a la interfaz de comandos, Linux ofrece escritorios y administradores de ventana. Ambas alternativas utilizan interfaces gráficas de usuario basadas en el Sistema X Windows desarrollado para UNIX por el consorcio Open Group.

Los alcances que Linux tiene hoy en día son enormes, día con día programadores de todo el mundo contribuyen para afinar y extender los horizontes del sistema operativo, y eso es otro aspecto por el cual hace a Linux una plataforma completa, sólida y segura, que no son un grupo de personas quienes hacen las correcciones y mejoras, si no son miles de programadores alrededor de mundo unidos a través de Internet.

Algunas de las distribuciones de Linux:

- Red Hat Linux (actualmente distribuyendo la versión Fedora)
- OpenLinux (Caldera)
- SuSE Linux
- Debian Linux
- LinuxPPC (versión Mac PowerPC)
- Turbo Linux (Pacific Hi-Tech)
- Slackware Linux Project
- Linux Mandrake

En este caso en particular de la propuesta cualquier distribución de Linux cumple con los requerimientos mínimos de sistema operativo que se va a utilizar en la implementación, ya que todo el software que se utilizará es 100% compatible con este sistema operativo.

3.1.2 Firewall iptables

Un Firewall es un equipo de hardware o software utilizado en las redes para prevenir algunos tipos de comunicaciones prohibidos por las políticas de red, las cuales se fundamentan en las necesidades del usuario, esto es, asegurar que todas las comunicaciones entre dicha red e Internet se realicen conforme a las políticas de seguridad de la organización que lo instala. Además, estos sistemas suelen incorporar elementos de privacidad, autentificación, etc.

IPtables es un sistema de firewall vinculado al kernel de linux que se ha extendido enormemente a partir del kernel 2.4 de este sistema operativo. Al igual que el anterior sistema ipchains, un firewall de iptables no es como un servidor que lo iniciamos o detenemos o que se pueda caer por un error de programación, iptables esta integrado con el kernel, es parte del sistema operativo. ¿Cómo se pone en marcha? Realmente lo que se hace es aplicar reglas. Para ellos se ejecuta el comando iptables, con el que añadimos, borramos, o creamos reglas. Por ello un firewall de iptables no es sino un simple script de shell en el que se van ejecutando las reglas de firewall.

IPtables es una estructura de tabla genérica para la definición de un conjunto de reglas. Cada regla dentro de una tabla IP tiene un sin número de clasificadores (IP a aceptar) y una acción objetivo(para que tiene permiso o que no tiene permiso).

En el listado L3.1 se muestran instrucciones las cuales establecen políticas para determinadas direcciones IP, así es como se definen las políticas de firewall en IPtables.

```
## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar

# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT
```

```
# A nuestra IP le dejamos todo
iptables -A INPUT -s 195.65.34.234 -j ACCEPT

# A un colega le dejamos entrar al mysql para que mantenga la BBDD
iptables -A INPUT -s 231.45.134.23 -p tcp --dport 3306 -j ACCEPT

# A un diseñador le dejamos usar el FTP
iptables -A INPUT -s 80.37.45.194 -p tcp -dport 20:21 -j ACCEPT

# El puerto 80 de www debe estar abierto, es un servidor web.
iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Y el resto, lo cerramos
iptables -A INPUT -p tcp --dport 20:21 -j DROP
iptables -A INPUT -p tcp --dport 3306 -j DROP
iptables -A INPUT -p tcp --dport 22 -j DROP
iptables -A INPUT -p tcp --dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Listado L3.1 Ejemplo de script de IPtables

3.2 Capa Cliente

Dentro de esta capa se consideran varias cosas , como es un navegador de Internet o puede ser una aplicación cliente el cual haga alguna especie de petición hacia algún servidor (que seria parte de la capa de presentación) y éste resuelva todo o tenga que enviárselo a otro servidor(capa de lógica y/o capa de datos).

Hablando de un navegador, se considera como una interfaz de usuario universal. Dentro de sus funciones están la petición de las páginas Web, la representación adecuada de sus contenidos y la gestión de los posibles errores que se produzcan.

Para todo esto, los fabricantes de navegadores les han dotado de posibilidades de ejecución de programas de tipo script, con modelos de objetos que permiten manipular los contenidos de los documentos. Estos lenguajes de programación son VBScript, JScript (ambas de Microsoft) y JavaScript (de Netscape), y proporcionan las soluciones llamadas del lado del cliente, client side y permiten realizar validaciones de datos recogidos en las páginas antes de enviarlos al servidor y proporcionan un alto grado de interacción con el usuario dentro del documento.

Otras de las posibilidades de los navegadores es la gestión del llamado HTML dinámico (Dinamic HTML, DHTML). Éste está compuesto de HTML, hojas de estilo en cascada, (Cascade Style Sheets, CSS), modelo de objetos y scripts de programación que permiten formatear y posicionar

correctamente los distintos elementos HTML de las páginas Web, permitiendo un mayor control sobre la visualización de las páginas.

En esta línea, los navegadores han ido un poco más allá y permiten la visualización de documentos XML (eXtensible Markup Language) después de haber sido transformado adecuadamente a HTML por las hojas de estilo extensibles (eXtensible Style Sheets, XSL). De esta manera se elige visualizar ciertos elementos y otros no, dependiendo de las circunstancias.

Además, los navegadores permiten la ejecución de aplicaciones dentro de los documentos mostrados. Las dos posibilidades más populares son la tecnología ActiveX y los Applets Java. Los Applets Java son pequeños programas que se descargan del servidor Web y se ejecutan en la JVM (Java Virtual Machine) del navegador.

El compilador Java únicamente genera el denominado ByteCode. Este código es un código intermedio entre el lenguaje máquina de procesador y Java. Evidentemente este código no es ejecutable por si mismo en ninguna plataforma hardware, pues no se corresponde con el lenguaje de ninguno de los procesadores que actualmente se conocen. Entonces, para ejecutar una aplicación Java es necesario disponer de un mecanismo que permita ejecutar el ByteCode. Este mecanismo es la denominada Máquina Virtual de Java (JVM). En cada plataforma de sistema operativo existe una máquina virtual específica. Así que cuando el ByteCode llega a la máquina virtual, esta lo interpreta pasándolo a código máquina de procesador donde se este trabajando, así como ejecutando las instrucciones en lenguaje maquina que se derive de la aplicación Java. De este modo, cuando el mismo ByteCode llega a diferentes plataformas, este se ejecutara de forma correcta, pues en cada una de esas plataformas existirá la maquina virtual adecuada. Con este mecanismo se consigue la famosa multiplataforma de Java, en la que con solo codificar una vez, podemos ejecutar en varias plataformas.

Éste es el funcionamiento general de la maquina virtual, en realidad tienes otras funciones, pero en nuestro caso de propuesta con productos software libre es esencial que el navegador de Internet que se vaya a usar, cuente con una JVM. Actualmente la mayoría (comerciales y no comerciales), si no es que casi todos soportan esta función.

3.2.1 Mozilla FireFox

Mozilla fue el nombre original para el producto que se vino llamando Netscape Navigator, y después Netscape Communicator.

Después, Netscape Communications Corporation s' vino a poner una mascota parecida a un dinosaurio. Hoy en día usamos el nombre de Mozilla como un término genérico referenciado al cliente de Internet desarrollado por un proyecto de software libre.

Mozilla es un conjunto de tecnologías, y mozilla.org es un grupo de personas que coordinan el proyecto.

El proyecto Mozilla esta compuesto actualmente de varios software's, los cuales se siguen mejorando estos son:

- FireFox que es un Web browser mejorado, el cual cuenta con muchos plug-in's que permite la ejecución de archivos en especifico.
- Thunderbird es un cliente de e-mail, que hace fácil y rápido el manejo de correos electrónicos
- Mozilla Suite que esta compuesto de herramientas parecidas a las listadas anteriormente pero incorpora un cliente de Chat, así como un editor HTML.
- Bugzilla es un software de seguimiento de bug's(errores de programación) , fue diseñada para ayudar a los equipos a desarrollar software
- Camino es un Web browser optimizado para Mac OS X
- Calendar Project es una aplicación de calendario

FireFox no reinvento el browser, pero provee mejoras técnicas que hacen al buscador mas rápido, seguro y fácil. FireFox incluye controles built-in para bloquear las molestas ventanas popups (ventanas que son llamadas dentro de un sitio sin que el cliente lo solicite).

Una característica importante de FireFox es el abrir muchas ventanas dentro de una misma ventana y rápidamente moverse entre páginas solo pulsando la pestaña correspondiente a la página, esta caracteritica evita que se tenga que abrir muchas instancias de browser, donde cada una requiere recursos del sistema.

Otra buena característica es la implementación RSS reader, que en otras palabras es permitir de forma automática actualizaciones de sitios de noticias a los cuales se ha suscrito, esta información es presentada y filtrada de tal manera que solo muestra lo que se desea ver.

Firefox es seguro, y esto se debe a que no soporta VBscript y controles ActiveX, que son fuente de ataques y vulnerabilidades. FireFox cuando abre una página segura, el FireFox pone subrayado en color la dirección (URL) y con un icono en forma de candado, al seleccionar con el ratón despliega la información de seguridad, y divide la URL de tal manera que una parte de la dirección la deja descubierta y la otra protegida contra cualquier intruso. En IE esto no pasa, si llega a detectar alguna vulnerabilidad, marca toda la pagina como segura o insegura.

FireFox posee una búsqueda inteligente (mas que inteligente muy fácil de usar y cómodo sin necesidad de instalar programas alternos en la máquina); incorpora el poder leer encabezados de noticias, actualizaciones de sitios favoritos, etc.

Firefox tiene 100% soporte con Java, Real player, productos de la serie Macromedia, entre otros.

3.3 Capa Presentación

Es en esta etapa donde se sitúa la arquitectura necesaria para atender las peticiones de los clientes, en esta etapa también se incluirá la parte de seguridad, es decir filtros y firewall, así como servidor Web y servidor de aplicaciones, alcances del producto propuesto y ventajas.

La capa de presentación requiere manejar mecanismos para recibir diferentes tipos de requerimientos, para tipos de procesamiento. Algunos de estos requerimientos son simplemente re-enviados a la correspondiente componente que maneje este dato, mientras que otros requerimientos son modificados, auditados o descomprimidos antes de continuar el procesamiento.

Por ejemplo, cuando un usuario solicita una página Web, ésta requiere realizar estas verificaciones previas antes de realizar el procesamiento principal de la página, ejemplo:

- ¿El cliente ha sido autentificado?
- ¿El cliente tiene una sesión válida dentro del sistema?
- ¿La dirección IP del cliente, pertenece a una red confiable?
- ¿El path requerido no cumple con alguna restricción?
- ¿El sistema soporta la versión del browser del usuario?

Mientras que algunos de estas validaciones son test del tipo SI o NO, otros requieren una modificación de la información de entrada a la página. La solución clásica a este problema es el uso de variadas condiciones del tipo if/else. Sin embargo, una solución de este tipo nos lleva a un código frágil y con un estilo de programación basado en "copiar y pegar".

La solución planteada por este patrón es crear filtros que procesen los servicios comunes en una forma estándar, de tal forma que no requiera efectuar cambios en el núcleo del código que procese los requerimientos. Los filtros interceptan los datos de entrada y las respuestas de salida, permitiendo un pre-procesamiento y un post-procesamiento. Además permiten agregar o eliminar filtros, sin requerir una modificación del código existente.

El servidor Web es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición.

Para abrir una página Web en un navegador, normalmente se teclea el correspondiente URL o se pica en el hiperenlace correcto. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza la página Web en su sistema de archivos y la envía de vuelta al navegador que la solicitó (ver figura F3.4).

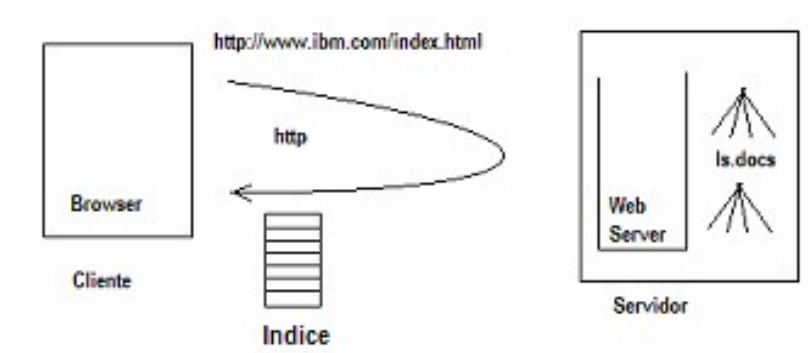

Figura F3.4 Ciclo de una petición http

Un servidor Web sólo sirve contenido estático. Para generar contenido dinámico es necesario otro componente, por ejemplo un servidor de aplicaciones o una interfaz (CGI o similar) hacia Perl, PHP u otro lenguaje script.

3.3.1 Mecanismo de hospedaje virtual de sitios Web

El mecanismo de hospedaje virtual de sitios Web (Web virtual hosting) consiste en la simulación de que existen varios hosts con sus respectivos sitios Web a través de un solo servidor Web. Cada uno de estos hosts virtuales se identifica con un nombre de dominio y de acuerdo a este son accedidos los documentos asociados al mismo y que en realidad son proporcionados por un único servidor Web.

Existen dos variantes de este mecanismo: la primera es aquella donde se asocia a cada host virtual una dirección IP diferente y la segunda es cuando se utiliza la misma dirección IP para todos los hosts. La última es la más elegante y conveniente sobre todo en los casos en que el número de hosts tienda a crecer

3.3.2 Servidor Web Apache

El servidor Web Apache es el servidor Web más difundido y utilizado en Internet, según las estadísticas de servidores Web de Netcraft (www.netcraft.com/survey/), el 63.7% de los sitios activos de Internet utilizan Apache, contra un 27.39% que utilizan el servidor de Microsoft y el resto repartido entre otros, esto en número de servidores está traducido como más de 7 millones de servidores que sirven poco más de 18 millones de sitios Web.

Tal vez lo que hace más atractivo a Apache es su alta estabilidad, seguridad y facilidad de expansión, aparte de su costo(es gratis), ya que es software libre. Quien quiera puede descargarlo de www.apache.org, y no tendrá que pagar costosas licencias de uso, además se dispone de su código fuente, para adecuarlo a alguna necesidad que se tenga.

El servidor Web Apache se encuentra disponible para una gran cantidad de sistemas operativos. La última versión (v 2.2 hasta la última fecha de edición de este trabajo) ha introducido cambios significativos y ha mejorado notablemente el desempeño y manejo de este servidor bajo Windows.

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

Entre las características principales del Apache se encuentran:

- Es un servidor Web potente, flexible y ajustado al HTTP/1.1
- Es altamente configurable y extensible.
- Es ajustable a través de la definición de módulos empleando sus propias API's.
- Provee todo su código fuente de forma libre y se distribuye bajo una licencia no restrictiva.
- Se ejecuta en diversas plataformas operativas tales como: Windows 9x/NT/XP/200X, Macintosh, Novell NetWare, OS/2, Linux y la mayoría de los Unix existentes: IRIX, Solaris, HPUX, SCO, FreeBSD, NetBSD, AIX, Digital Unix, etc.
- Se desarrolla de forma acelerada estimulando la retroalimentación desde sus usuarios a través de nuevas ideas, reportes de errores y parches.
- Apache significa ``A PAtCHy sErver", o sea se basa en un código y un conjunto de ficheros ``parches". Otros desarrolladores relacionan su nombre con el de las tribus nativas americanas de Apaches.
- Implementa muchas posibilidades frecuentemente demandadas, tales como:
    - Bases de datos DBM para autenticación. Permiten establecer fácilmente la protección de documentos a través de passwords para una gran cantidad de usuarios sin dañar el funcionamiento del servidor.
    - Respuestas adaptables a los errores o problemas. Tiene la capacidad de definir archivos o scripts de tipo CGI que respondan ante la ocurrencia de errores internos o en las solicitudes realizadas.
    - Directiva para definir múltiples índices. Se utiliza cuando se solicitan directorios por parte de los clientes a partir del cual buscan y devuelven un documento índice cuyo nombre puede ser por ejemplo: index.html, index.cgi o default.html.
    - Ilimitadas y flexibles posibilidades de redireccionamiento y definición de alias para los URL. Apache no tiene un límite establecido para definir alias y redireccionamientos que son declarados en sus archivos de configuración.
    - Negociación del contenido de las respuestas. Apache es capaz de ofrecer la mejor representación de la información accedida de acuerdo con las capacidades del cliente solicitante.
    - Soporte de hosts virtuales. Es la habilidad del servidor de distinguir entre los pedidos hechos a diferentes direcciones IP o nombres de dominio definidos en la misma máquina.

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

El paquete de la distribución Red Hat que contiene la implementación del servidor Apache para Linux se nombra apache. También se dispone de un manual del mismo en el paquete apache-manual. Existe una aplicación con interfaz gráfica en el paquete apacheconf que permite configurar al Apache con las limitaciones propias de dichas interfaces.

El Apache en Red Hat se ejecuta a través de un proceso daemon llamado httpd que se manipula utilizando el script de inicio del mismo nombre.

En la propuesta este servidor va a ser usado para atender las peticiones del cliente y delegarlas hacia el recurso correspondiente. Basándonos en una arquitectura J2EE, el servidor Apache atiende la petición y la enviara hacia el servidor de aplicaciones correspondiente, o en su caso particular hacia un contenedor, el contenedor hacia el recurso final.

3.3.3 Jakarta Tomcat Web Container

El primer contenedor de Servlets fue el Sun Microsystem's Java Web Server (JWS). En un principio parecía ser el más ajustable incluso que los propios servidores comerciales, pero no cumplía con las demandas empresariales reales, por carecer de funciones administrativas. Eso era a la novedad de Java y de que el concepto de Servlets había sido recientemente introducido. El mayor éxito del JWS es que puso a los Servlets a descubierto ante los ojos del mundo listo para ser explotado.

En 1996, empezaron a surgir los contenedores de Java Servlets y a hacerse más populares. El Apache JServ y CERN/W3C's Jigsaw fueron dos de esos contenedores. Fueron seguidos por mas, el Jetty, el Locomotiva Application Server, Enhydra, y muchos otros, al mismo tiempo, contenedores comerciales empezaron a surgir siguiendo el estándar Servlet, algunos de estos fueron WebLogic's Tengah y ATG's Dynamo.

En 1997, Sun lanzo su primera versión del Java Servlet Development Kit (JSDK). El JSDK fue un pequeño contenedor Servlet que soportaba JavaServer Pages (JSP) y le fue integrado el servidor Web HTTP 1.0. En un esfuerzo por proveer una implementación para desarrollo de Servlets, Sun lo hizo libre y descargable para todo mundo esperando experimentar con el nuevo estándar Java del lado del servidor.

En la primera mitad de 1998, Sun anuncio su nueva especificación JSP, que incluía el API Servlet java y permitía un rápido desarrollo de aplicaciones de contenido dinámico Web.

Tomcat fue creado por James Duncan Davidson (quien rescribió el JSDK, ahora llamado JSWDK), lo escribió basándose sobre los Servlet y la idea JSP. Tomcat evolucionó rápidamente y creció. Cuando las especificaciones llegaron a ser ricas, por varias razones, se decidió abrir el código JSWDK, con el fin de revelar como los Servlets y JSP operaban.

Poco después se decidió donar el proyecto Tomcat al Grupo Apache Software Foundation, con el fin de que se hiciera lo más acorde a la especificación Servlet, haciéndose la implementación de referencia para la especificación Servlet de Java.

Aunque Tomcat es un contenedor que puede trabajar solo, incluso como Servidor Web, carece de características como por ejemplo que no tiene una lista de módulos opcionales a usar(lenguajes script, etc.), lo que lo haría más lento. Es por ello que en esta propuesta se usa como contenedor Web de aplicaciones, donde trabajará en conjunto con el servidor Web Apache, usando un modulo llamado mod_jk2.

Una de las capacidades que tiene Tomcat es de que las aplicaciones Web declara que recursos son accesibles por que grupos de usuarios en el archivo descriptor web.xml.

Características principales de Tomcat:

- Estricta adherencia a la especificación de SUN JSP/Servlet.- Tomcat ha venido siendo la referencia de implementación, esto es el implementar los Servlets y JSP tanto como se pueda.
- Interoperabilidad.- existen muchos servidores Web operando en el mercado, Tomcat está hecho para interactuar con ellos (en algunos casos de forma mínima)
- Facilidad de efectuar modificación.- Internet esta constantemente evolucionando y su comunidad de usuarios crece. Esto permite ampliar los escenarios de aplicaciones Web. Como resultado, existe una constante de nuevos requerimientos que impactan en la arquitectura de Tomcat. Entonces, la arquitectura es adaptable a estos cambios.
- Rendimiento.- el rendimiento esta siendo incrementado importantemente en los ambientes de hoy en día donde aplicaciones Web, demanda de comercio electrónico, demandan alta nivel de rendimiento en el motor de Servlets. Entonces, la arquitectura es ligera para los usuarios del motor Servlet. En otras palabras la arquitectura es robusta para poder soportar las necesidades del mercado.
- Escalabilidad.- la arquitectura escala de acuerdo al número de solicitudes por usuario sin la degradación en el desempeño
- Alta disponibilidad.- en el mundo de los e-business, las aplicaciones Web proveen alta disponibilidad de servicio a los clientes. Entonces esto exige que el motor nunca deje de funcionar. Entonces la arquitectura que día a día mejora soporta escenarios que resultan ser altamente disponibles.
- Seguridad.- La seguridad Web es de suma importancia para todas las aplicaciones, como motor de Servlets, Tomcat provee facilidades para conectarse a varios mecanismos de autentificación con grados de variación de seguridad. Inclusive provee la infraestructura de autentificación de usuarios e integridad de datos envueltos en la comunicación.

3.4 Capa lógica de negocio

El comportamiento de la aplicación es definido por los componentes que modelan la lógica de negocio. Estos componentes reciben las acciones a realizar a través de la capa de presentación, y llevan a cabo las tareas necesarias utilizando la capa de datos para manipular la información del sistema. Tener la lógica de negocio separada del resto del sistema también permite una integración más sencilla y eficaz con sistemas externos, ya que la misma lógica utilizada por la capa de presentación es accedida desde procesos automáticos que intercambian información con los mismos.

El nivel de lógica de negocio es la porción de la arquitectura realizada por componentes utilizados para crear y garantizar las reglas de clientes, productos y de negocio.

Las funciones fundamentales de esta capa y de los objetos que se generen dentro son las siguientes:

- Realizar la validación de los datos introducidos por el usuario, tanto física (numérico, fecha, etc.) como lógica (importe menor, que saldo, etc.)
- Ejecutar la petición realizada por el usuario. Para ello podrá valerse de transacciones contra Sistemas Host, consultas a la base de datos, consultas a proveedores de contenidos, etc.
- Generar los objetos que utilizaran las vistas para mostrar los resultados obtenidos
- Garantizar la navegación y el flujo de pantallas correcto

3.4.1 JBoss

JBoss ha marcado un camino de innovaciones Java diseñado con soporte total del estándar J2EE. Inicialmente empezó en 1999 como un contenedor software libre de EJB, partir de la versión 2.X se convirtió en un servidor basado en J2EE (donde se distribuían todos los módulos que comprenden el J2EE, haciendo uso de proyectos de software libre como Tomcat, para proveer un contenedor Web). JBoss para la versión 3.X (que es la que se distribuye actualmente) se convirtió en un marco de trabajo muy completo en donde se construyen aplicaciones, basadas en JMX(Java Management Extensions) y en una arquitectura orientada al servicio (Service Oriented Architecture (SOA).

Fue desarrollado en código abierto, JBoss no sólo es favorecido por desarrolladores avanzados, si no también por desarrolladores novatos en Java ampliando los alcances del producto. JBoss es distribuido bajo la licencia LGPL, es el numero uno en descargas con más de 5 millones de descargas, también es el número 3 (servidor de aplicaciones) con el 25% del mercado actual, integra servicios de seguridad y permite uso de objetos en forma remota.

Como ya hemos visto en un Servidores de Aplicaciones Java destacan 2 partes: un "Servlet Engine" y un "EJB Engine", dentro del "Servlet Engine" se ejecutan exclusivamente las clásicas aplicaciones de Servidor (JSP's y Servlets), mientras el "EJB Engine (Contenedor)" es reservado para aplicaciones desarrolladas alrededor de EJB's.

JBoss trabaja de la mano con Tomcat(con unos pequeños detalles incluidos por el JBoss group, es este contenedor Web con el que actualmente se distribuye JBoss) ; lo anterior no restringe a JBoss para operar con otro contenedor Web como ServletExec, la única diferencia de utilizar aquellos contenedores Web, será en coordinación y configuración entre JBoss y el contenedor Web.

De forma grafica se ilustran los contenedores en la figura F3.5:

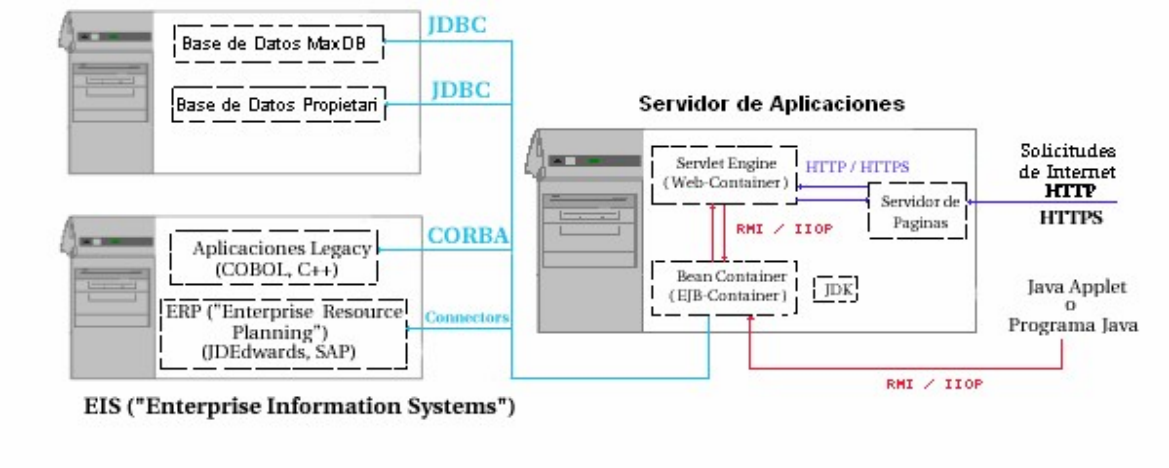

Figura F3.5 Contenedores del servidor de aplicaciones Jboss

En párrafos anteriores se habló del Java Management eXtensions (JMX), para tocar ese punto es necesario hablar un poco de este aspecto. JMX ha sido desarrollado por SUN Microsystems durante muchos años, aunque formalmente es llamado JMAPI. JMX permite un manejo centralizado de beans o MBeans, que actúan como ćlips ”para aplicaciones, componentes, o recursos en una red distribuida. Esta funcionalidad es proveída por un servidor de beans. En un modelo de arquitectura JMX, el servidor de beans (o MBeans) viene siendo la espina dorsal del servidor donde todos los servidores componentes están conectados y otros servidores de beans externos con los que interactúa.

La arquitectura de JBoss esta formada por lo siguiente:

- El microkernel.- Es la base de JBoss donde usa una implementación JMX, provee un modelo por componentes bastante ligero con total soporte de desplegado a través de la red(permite tomar componentes de otras partes e ingresarla a la arquitectura), posee características avanzadas para la carga de clases y manejo del ciclo de vida de los beans
- Soporte para una arquitectura orientada al servicio.- Se refiere a que con base a las capacidades que tiene el microkernel, JBoss provee servicios, cada servicio es empaquetado en algo conocido como Service Archive (SAR). Los servicios son un monitor transaccional, un servicio de mensajería, servicio de correo, base de datos, o algún servicio de conexión. Se pueden agregar más servicio personalizados, extendiendo las capacidades de JBoss

- Nivel de aspecto.- el nivel de aspecto está dentro de un término llamado interceptores. Los interceptores permiten al sistema agregar transparentemente el comportamiento proveído por los servicios dentro de cualquier objeto.
- Nivel de aplicación.- aquí es donde las aplicaciones viven (en donde Tomcat forma parte de este nivel, dentro de JBoss, el servidor Web Apache es quien sirve la peticion).

3.5 Capa de Datos

Es en esta capa donde prácticamente residen los sistemas de almacenamiento de información como los Sistemas Base de Datos Relacionales, Sistemas de Base de Datos Orientados a Objetos, Sistema Manejadores de Archivos, etc.

En el caso de los sistemas de bases de datos, existen muchas alternativas tanto en el ámbito comercial como en el de software libre para el desarrollo de sistemas y aplicaciones de información. Las bases de datos se han ido desarrollando y evolucionando desde las primeras aplicaciones que en realidad empleaban archivos con una estructura jerárquica, hasta los sistemas de administración de bases de datos DBMS con capacidades para el manejo y procesamiento tanto de estructuras relacionales clásicas como de orientación a objetos y datos complejos, así como del procesamiento de millones de registros, miles de usuarios simultáneos y terabytes de información.

3.5.1 MaxDB

En el mundo del software libre hasta el momento, existen del conocimiento de la mayoría de los programadores y desarrolladores alrededor del mundo, dos sistemas de bases de datos, PostgreSQL y MySQL. Este último con mucho, el más popular lo que se comprueba con las miles de instalaciones en diferentes lugares alrededor del mundo.

MySQL se caracteriza por la rapidez y facilidad de uso lo que permite que para una inmensa mayoría de aplicaciones, sirva perfectamente para el almacenamiento y recuperación de información sin las complicaciones ni costos de los sistemas comerciales como Oracle, SQL Server, Informix, DB2, Sybase y otros.

Sin embargo, existe otro sistema de bases de datos relacionales que hereda toda la experiencia y conocimiento de los sistemas de bases de datos creadas en los años 70´s. Este sistema de bases de datos se llama en la actualidad MaxDB. MaxDB representa una evolución sostenida desde sus inicios en la Universidad Técnica de Berlín hasta el desarrollo de la base de datos SAP DB desarrollada y mantenida por SAP AG.

MaxDB se diseñó, con la capacidad para soportar un muy alto nivel de transacciones y miles de usuarios concurrentes, así como el procesamiento de algoritmos muy complejos de una forma eficiente.

En 1970, existía un proyecto de investigación en la Universidad Técnica de Berlín titulado "Bases de datos distribuidas en minicomputadores." Casi al final de los años 70´s, la empresa Nixdorf Computer AG descubrió este proyecto y lo patrocinó motivado por el hecho de que prometía un gran potencial para el desarrollo de un poderoso sistema de bases de datos relacionales. El proyecto de investigación pronto se convirtió en un producto comercial y por lo tanto, en una más de las soluciones de software que la empresa ofrecía para el procesamiento de datos y transacciones. Durante la década de los años 80´s, el sistema de bases de datos fue comercializado por Nixdorf bajo el nombre DDB/4 y la plataforma elegida para el desarrollo y mantenimiento del sistema fue el sistema operativo Unix debido a su estabilidad y robustez.

En el año de 1989 Nixdorf Computer fue adquirida por Siemens AG y el equipo de desarrollo formó parte de la nueva subsidiaria de Siemens Nixdorf Informations Systeme AG (SIN). Como consecuencia de esta transacción comercial, el producto se renombró como ADABAS D.

En 1993, el sistema fue adaptado y modificado por Software AG, para cumplir con los requerimientos de los sistemas administrativos ERP SAP/R3 y además se celebró un acuerdo de licenciamiento conjunto con SAP.

A finales de 1997, Software AG se retiró de la comercialización de ADABAS D para R/3 y como consecuencia SAP AG retomó el soporte para la base instalada de clientes que ya tenían ambos productos. El nombre se modificó entonces a SAP DB y SAP AG inició entonces una fuerte inversión de su propia versión de DBMS.

SAP AG decidió en el año 2000, transferir su sistema de administración de bases de datos relacionales SAP DB hacia un modelo de código abierto u Open Source. Este sistema de bases de datos ha sido ofrecido a todos los clientes y usuarios de las soluciones mySAP (suite de herramientas de solución de SAP) desde el año 2000 y el código fuente esta disponible para la comunidad de software libre desde finales de Abril del año 2001.

SAP DB (ahora MaxDB) se distribuye con la licencia GNU GPL, se eligió este método de licenciamiento con el objeto de que el proceso fuera lo más transparente claro y simple posible. Esta iniciativa fue un paso muy importante para la empresa y para los usuarios por las siguientes razones:

- Estaba orientada a crear y fomentar una comunidad a nivel mundial alrededor del producto.
- Demostrar que los sistemas profesionales de bases de datos no tienen porque ser caros.
- Obtener mayor retroalimentación de áreas y comunidades fuera de los equipos de desarrollo de SAP y sus clientes alrededor del mundo para mejorar el producto.

Una razón adicional para que SAP AG decidiera heredar el código fuente a la comunidad de software libre es la de mantenerse tecnológicamente independiente de los fabricantes de sistemas de bases de datos relacionales lo más posible.

En mayo del 2003, una asociación tecnológica fue formada entre MySQL AB y SAP AG. Esa asociación encarga a MySQL AB a desarrollar SAP DB. En agosto del 2003, SAP DB fue

renombrada a MAX DB por MySQL AB. Algunos de los usuarios de MaxDB son: Toyota, Intel, DaimlerChrysler, Braun-Gillette, Bayer, Colgate, Yamaha, y el servicio postal Alemán.

La avanzada arquitectura de MaxDB le permite generar un alto desempeño, escalabilidad y robustez los cuales son elementos indispensables y de gran valor en ambientes complejos conformados por miles de usuarios simultáneos, bases de datos que rondan en los terabytes de información.

Los avances y mejoras que SAP ha incorporado a su base de datos incluyen tecnología para el procesamiento eficiente de documentos sin formato o que se encuentren parcialmente estructurados que son comunes en las modernas aplicaciones de Internet o Intranet.

La arquitectura de MaxDB le permite (y le obliga) operar de forma no interrumpida y llevar a cabo tareas de mantenimiento, como modificaciones a la configuración de instancias, ampliación de áreas de datos o logs, respaldo y restauración de datos, así como la creación de índices y tablas sin necesidad de desactivar a los usuarios o dar de baja el sistema, lo que lo habilita para una operación normal aun cuando se requieran trabajos de afinación, prevención, mantenimiento, respaldo o recuperación del sistema. Esta capacidad para llevar a cabo actividades multitarea permiten disminuir enormemente el tiempo que los usuarios requieren para que sus tareas se concluyan y contribuyen a una operación continúa del sistema sin tiempos muertos por mantenimiento.

Aun cuando existen otros sistemas de bases de datos en el ambiente Linux, MaxDB se distingue de los otros como son PostgreSQL o MySQL en rendimiento y eficiencia. MaxDB se diseñó desde el principio para aplicaciones OLTP que involucren un gran número de usuarios y transacciones de forma simultánea. MaxDB por lo tanto puede soportar los complejos requerimientos de las aplicaciones SQL de los sistemas SAP y de muchas otras aplicaciones con necesidades de procesamiento de muy grandes cantidades de información.

MaxDB está desarrollado, compilado y optimizado para operar en diferentes plataformas, lo que lo hace uno de los DBMS más versátiles. Las plataformas en las que MaxDB corre son las siguientes:

- HP Compaq Tru64 (Alpha)
- IBM AIX (PowerPC)
- SUN Solaris (SPARC)
- HP-UX (HP-PA)
- Linux (Intel 32 y 64 bits)
- AMD 32 y 64 bits
- Siemens Reliant Unix (MIPS)
- Windows NT/2000/2003 (Intel)

3.6 Metodologías para la Creación de Software de Alto Nivel

Una vez planteada la arquitectura basada en software libre y conocidos los estándares abiertos necesarios para la creación de sistemas computacionales de escala empresarial es necesario hablar ahora del modelo que sigue la mejora continua de estos hasta crearlos con una excelente calidad.

3.6.1 Capability Maturity Model (CMM) for Software

Aunque ingenieros de software y directores saben a menudo sus problemas con gran detalle, estos ingenieros están en desacuerdo sobre que mejoras son más importantes. Sin una estrategia de organización para mejoras, es difícil de lograr el consenso entre administración y el personal profesional sobre que actividades de mejora se harán primero. Para lograr resultados óptimos del proceso de mejora, es necesario diseñar un camino evolutivo que aumenta la madurez del proceso de software de una organización en etapas.

El modelo de madurez de capacidad (CMM) para software provee organizaciones de software guía sobre como obtener control de sus procesos para desarrollo y mantenimiento de software y como evolucionar hacia una cultura de la ingeniería de software y excelencia de administración. El CMM fue diseñado para guiar organizaciones de software en la selección de estrategias de mejora de procesos por determinada madurez de procesos actual e identificando los pocos aspectos críticos de calidad de software y mejora de procesos, enfocándose sobre un conjunto limitado de actividades y trabajando agresivamente para lograrlas, una organización puede mejorar constantemente su proceso amplio de organización de software para permitir ganancias continuas en la capacidad del proceso de software.

La estructura en etapas del CMM esta basada sobre principios de calidad de producto que ha existido durante los últimos 60 años, Walter Shewart, promulgó los principios de control de calidad. Estos principios han sido adoptados por el SEI y afinados con el tiempo, en un marco de trabajo de madurez que establece un proyecto de administración e ingeniería para control en cantidad de los procesos de software, que es la base de un proceso continuo de mejora.

Un nivel de madurez se define como una cuesta evolutiva bien definida hacia lograr un proceso maduro de software. Cada nivel de maduración provee una capa en la fundamento para un proceso de mejora continua. Cada nivel se comprende de un conjunto de metas de ese proceso, que cuando se satisfacen, estabilizan un componente importante del proceso de software.

El proceso de mejora contínua esta basado sobre pequeños pasos evolutivos antes que las innovaciones revolucionarias. El CMM provee un marco de trabajo para organizar estos pasos evolutivos en 5 niveles de maduración para el proceso de mejora contínua. Estos 5 niveles de madurez definen una escala ordinal para medir la madurez de un proceso de maduración de organizaciones de software y para evaluar su capacidad de proceso de software. Estos niveles ayudan a dar prioridad los esfuerzos de mejora.

Los 5 niveles en como se organiza el CMM se muestra en la figura F3.6.

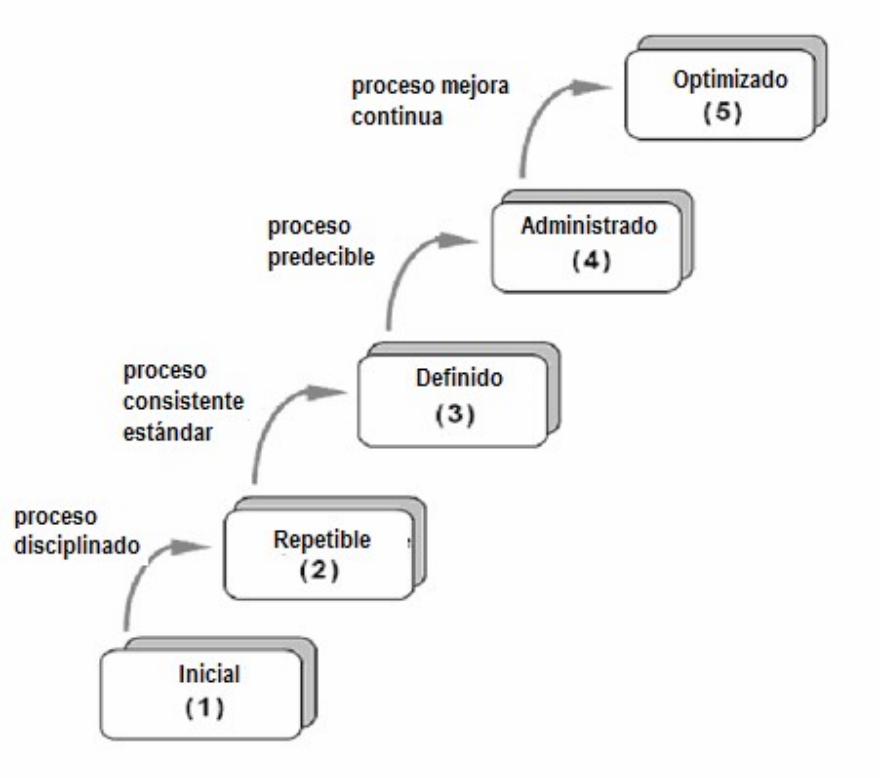

Figura F3.6 Niveles de organización de CMM

1) Inicial.- En esta etapa el proceso de software es difícil y complejo, se considera incluso que esta en un lapso caótico, porque pocos procesos están definidos, y el completar algo depende del esfuerzo individual.
2) Repetible.- Un proyecto básico de administración de procesos es establecido para seguir costos, itinerario, y funcionalidad. La disciplina de proceso necesaria esta en su lugar para repetir metas tempranas sobre proyectos con aplicaciones similares
3) Definido.- El proceso de software para ambas, administración y actividades de ingeniería, está documentado, estandarizado, e integrado dentro de un proceso de software estándar para la organización. Todos los proyectos usan una versión aprobada de proceso estándar para el desarrollo y administración de software.
4) Manejado.- Medidas detalladas del proceso de software y calidad de producto son colectados. Ambos, el proceso de software y los productos, son entendidos y controlados.
5) Optimizando.- el proceso continuo de mejora esta habilitado, donde hay innovación de ideas y tecnologías

El CMM es un modelo descriptivo, para plantear los atributos esenciales que se espera para caracterizar una organización en un nivel de madurez en particular. Es un modelo normativo en el sentido que las prácticas detalladas caracterizan los tipos normales de la conducta que se esperaría en una organización haciendo proyectos a gran escala.

3.6.2 Misión Crítica

En el mundo real existen aplicaciones las cuales tienen determinado alcance, estas son llamadas NAC (B2C "business to customer ," negocio a clientes) donde su alcance está orientado a clientes y las NAN (B2B "business to business ," negocio a negocio) están orientados a negocios desde otro negocio. Hoy en día muchas de esas aplicaciones no cumplen con las expectativas de disponibilidad y servicio permanente, muchos otros problemas son la falta de medio de pago confiable (para el caso de las NAC, ya que interactúan directamente con el cliente).

En la actualidad muchas empresas interesadas en el NAN tienen ya alguna forma de transacción electrónica con sus proveedores y grandes clientes. Estas transacciónes son en forma de simples mensajes de correo electrónico o sofisticados sistemas de intercambio de bits. De cualquier manera, el proceso de establecer una relación confiable ya se ha cumplido en el mundo real. También están definidos las condiciones comerciales, precios y componentes (hardware) que permiten que el traslado a un medio electrónico sea más fácil. Lo que si exige, es que se utiliza la Internet como medio confiable para manejo de procesos de misión crítica.

Misión crítica se define como aquellos procesos que no se pueden parar en una empresa porque ocasionarían pérdidas incalculables. Como son la facturación de una empresa de venta al menudeo, o los controles en un molino de papel, el sistema de compra y venta de acciones en la bolsa de valores entre muchos otros.

En la medida en que estos procesos se implanten sobre Internet, se requiere un análisis profundo de los prestadores del servicio de punta a punta de las transacciones para garantizar lo que se hace con exactitud, pero mejor aún, con confiabilidad y disponibilidad.

Tecnológicamente, se requiere un perfecto funcionamiento en todos los eslabones de esta cadena de proceso. En cada una de las puntas (cliente y proveedor) hay una conexión a la gran nube de Internet. En estas puntas se evalúan sistemas redundantes de conexión, en especial con proveedores distintos y por medios distintos, de tal manera que la falla en una es suplida por la segunda. Es importante tener en cuenta que hay dos pasos en esta conexión, una de la empresa a la prestadora de servicios de Internet (ISP) y otra de la ISP a la Internet misma. En ambos tramos de la conexión garantiza redundancia.

Luego viene el tema del almacenamiento y procesamiento de la información como tal. En una punta se generará una orden de compra, que se convierte en pedido y/o orden de producción en el otro extremo. Aunque hay un mayor control sobre las transacciones que se generen de esta manera. Por lo general hay un "mensajero" en la mitad que hace el trabajo de llevar y traducir la información de un lado al otro. Si este "mensajero" no está disponible, no se realizan la transacción. En la actualidad hay empresas ofreciendo estos servicios de "mensajería" mediante centros electrónicos de negocios o hubs de información.

Otro componente de la transacción está en los sistemas de cada una de las empresas involucradas. Por lo general estos sistemas están diseñados para efectuar trabajos de mantenimiento nocturnamente o grandes procesos de consolidación. Con los nuevos

requerimientos, estas tareas deberán substituirse por otras que permitan un proceso continuo las 24 horas del día.

El establecimiento de un sistema confiable en todos los eslabones es complejo para las empresas, es aquí donde surgen las ASP, proveedoras de aplicaciones en Internet (aplication service providers). Algunas al interior de los grandes conglomerados, como respuesta a los altos costos del software, y otras como prestadoras de servicios a terceros que quieran participar. En estos casos también es de vital importancia la conexión al servidor donde residen las aplicaciones, de no acceder, no se podrían procesar las transacciones pertinentes de una empresa.

Se está avanzando de manera muy rápida para implementar nuevas tecnologías de comunicación a Internet, algunas implementaciones serán por empresas privadas, otras por empresas ya establecidas en el mercado como proveedoras de los servicios tradicionales, pero lo que si es claro es que con la infraestructura actual con la que cuentan muchas de las empresas consideradas grandes en este país, la Internet ya es tomada en serio para transacciones de misión crítica.

3.7 Arquitectura de software propuesta

Como se ha visto hasta este momento, n-capas no es una tecnología, sino una estrategia de uso de las tecnologías para crear un negocio a la vez que se obtiene del potencial de Internet.

La informática basada en n-capas no se refiere solamente al despliegue de clientes ligeros de bajo coste conectados a servidores de aplicaciones muy flexibles con balanceo de carga e integrados con bases de datos distribuidas existentes a lo largo de diferentes plataformas y localizaciones. En realidad tiene que ver con la aplicación de las tecnologías (arquitectura propuesta basada totalmente en software libre) relacionadas con desarrollos en n-capas para mejorar el conocimiento de los negocios y proveer un servicio de valor mediante la aplicación de esta avanzada tecnología como una solución para envolver oportunidades del mundo real. Para tener éxito en el futuro, una compañía establece simples canales de comercio electrónico. Las organizaciones de mañana tienen que reconfigurarse a sí mismas de forma continua, tanto interna como externamente, como un negocio electrónico "camaleónico," creando rápidamente relaciones y maximizando el rendimiento de una "empresa extendida". Los sistemas de información de negocio serán cada vez más adaptables, permitiendo la modernización regular de las estrategias de e-business y los modelos de negocio.

La arquitectura propuesta en la figura F3.7 se encuentra compuesta del software libre anteriormente explicado, en donde cada uno se encuentra ubicado de acuerdo a la funcionalidad que provee, siguiendo un arquitectura en n- capas.

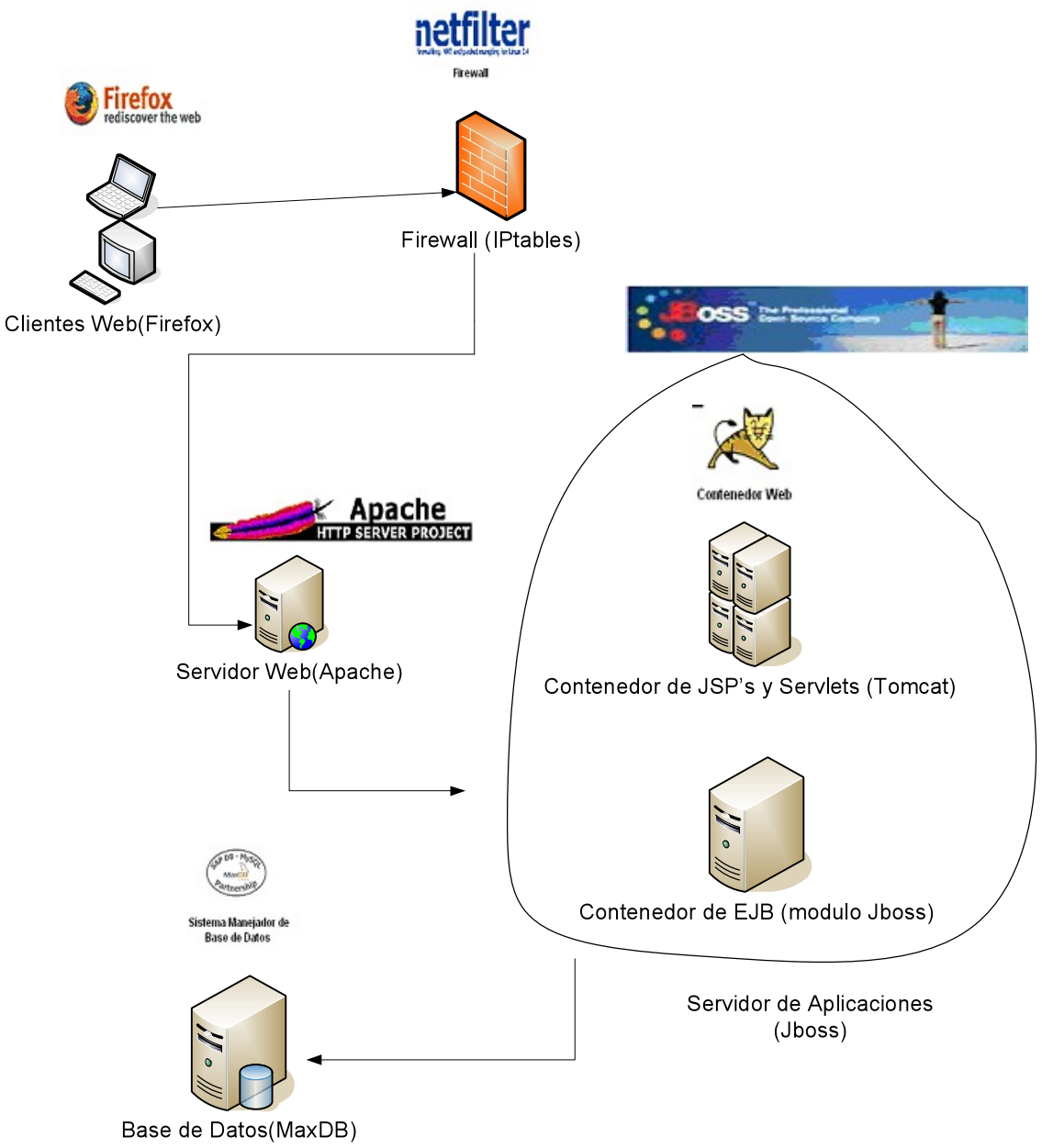

Figura F3.7 Arquitectura propuesta bajo el estandar J2EE

Capítulo 4

4 Aplicación de la arquitectura J2EE utilizando software libre en aplicaciones de misión crítica

4.1 Introducción a una solución empresarial

Hasta este punto se tiene ya una idea de una arquitectura corporativa propuesta siguiendo estándares abiertos, así como software que componen a esa arquitectura siguiendo el concepto de software libre.

En este capítulo se presentará un software que poniéndolo en funcionamiento con la arquitectura empresarial anteriormente vista, este trabaja con un índice de calidad muy alto siguiendo técnicas y estándares abiertos, este software se llama Compiere. Compiere se describira desde su aspecto técnico, como también de funcionalidad.

Compiere provee un integrado CRM (Customer Relationship Management), PRM (Partner Relationship Management), SCM (Supply Chain Management), ERP (Enterprise Resource Planning), OLAP (Online Analitical Processing). Está diseñado para ser hospedado en un ambiente Web, permitiendo opciones de instalación muy accesibles.

El ADD (Active Data Directory) que integra asegura una funcionalidad estable, así como una consistencia en su manejo.

Compiere está diseñado para cambiar según vaya evolucionando el negocio. En cualquier momento, clientes (incluso en producción) cambia la estructura de la información, ajustándose hacia las nuevas necesidades de información. En contraste, los sistemas tradicionales de ERP/CRM están ofreciendo seguimiento de cuentas, resultando en un hueco de información, que es rellenada por información derivada o cara y por puentes ineficientes.

Compiere provee múltiples vistas de la información negocio basada en el detalle de las transacciones actuales. Esta estructura permite un máximo de flexibilidad y fácil integración de información externa suplementaria. La información es simplemente presentada como vistas, o es cambiada rápidamente para alcanzar las necesidades del negocio. Compiere es software libre, no cuesta su licencia y puede ser descargado desde www.compiere.org.

4.2 Sistemas CRM, ERP, OLAP, PRM, SCM

CRM (Customer Relationship Management) es un conjunto de metodologías, software y capacidades de Internet que ayudan a una empresa a manejar las relaciones con el cliente de una manera mas organizada. Esto incluye todos los procesos en ventas, marketing, servicios que interactúan con el cliente. Por ejemplo, una empresa podría construir una base de datos acerca de sus clientes que describe las relaciones en el detalle suficiente para que administración, vendedores, gente dando servicio, o incluso el cliente puedan acceder información, alcanzando necesidades del cliente con planes de productos y ofrecimientos, recuerda a clientes sobre requerimientos de servicio, hace saber sobre que otros productos un cliente tiene comprado, etc.

ERP (Enterprise Resource Planning) es un software que tiene un total soporte para los procesos de negocios de una empresa. Típicamente consiste de módulos como marketing y ventas, servicios de campo, producción, control de inventario, distribución, recursos humanos, finanzas y contabilidad.

OLAP (Online Analitical Processing) es una herramienta que provee análisis de datos almacenados en una base de datos. Estas herramientas habilitan a los usuario a analizar en diferentes dimensiones de datos multidimensionales. Por ejemplo, provee series de tiempo y vistas de análisis de tendencias. OLAP incluso es considerado una minería de datos en algunos casos.

SCM (Supply Chain Management) es una combinación de arte y ciencia que permite proveer el camino de una compañía para encontrar los componentes primarios necesarios para hacer un producto o servicio. A continuación se enlistan los 5 componentes básicos de una SCM:

- Plan.- Esta es una estratégica para manejar todos los recursos que se necesitan para satisfacer las necesidades que el cliente demanda para un producto en especifico o servicio. La gran pieza de la planeación es desarrollar un conjunto de métricas para monitorear que la cadena de proveedores sea eficiente, que sus costos disminuyan, y entrega de alta calidad.
- Escoger-fuente.- Se define a los proveedores que cumplirán con lo prometido, y los servicios que se necesitan para crear el producto o servicio.
- Hacer.- Este es el paso de la manufactura, Agenda de actividades necesarias para la producción, prueba, empaquetamiento y preparación para entrega.
- Entrega.- esta es la parte que se refiere a la logística. Desarrolla una red de almacenes, escoge medio de entrega para enviar productos hacia los clientes y habilita un sistema de facturación para recepción de pagos
- Regreso.- El problema de la cadena de proveedores. Crea una red para recibir los productos defectuosos o productos regresados por clientes, también es soporte a clientes quienes tienen algún problema con los productos entregados.

PRM (Partner Relationship Management) es una estrategia de negocio para desarrollar y mejorar las relaciones entre compañías y su canal de socios proveedores. Un PRM es incluso considerado parte de un CRM donde sirve como el canal que controla las relaciones con los socios.

4.3 Compiere

4.3.1 ¿Que es Compiere?

Compiere te da la información correcta a las personas que lo requieran en el momento apropiado. Esto permite a las empresas adaptar rápidamente las condiciones cambiantes del mercado, ayudando a asegurar el éxito de la empresa en el cambiante mercado global.

Compiere es un software ERP y CRM que provee una vista a 360° de una compañía, sus clientes y vendedores. Por años sólo las compañías grandes habían sido beneficiadas por el uso de una aplicación ERP (esto se debe a los costos tan elevados de implantación y licencias para adquirir un software asi). Ahora, Compiere pone el poder comunicaciones consistentes (funcionalmente entre integrantes de una empresa) y tiempo de respuesta dentro de los alcances de una empresa ofreciendo un software sin costo de licencias y uso, adaptable y escalable, asegurando que una compañía cuente con una solución para sus necesidades.

El código de Compiere está disponible a cualquier persona o grupo que intente aprovecharlo para ajustar o extender la funcionalidad de Compiere a las necesidades del interesado. Compiere con su flexibilidad y su capacidad de actualización hace de el una solución ál dia "(con las ultimas actualizaciones de funcionalidad). El usuario de Compiere incluso tiene el acceso a una comunidad activa de software libre, con la cual se puede consultar diversos temas (técnicos y funcionales) con el fin de enriquecer el conocimiento en su uso, así como también se puede contactar mas de 100 empresas consultoras alrededor del mundo en caso de requerir asistencia (este servicio es opcional y tiene un costo que depende de las tarifas de cada empresa).

4.3.2 Aspecto técnico de Compiere

El servidor de Compiere está basado en tecnología J2EE, siguiendo esto Compiere usa como servidor de aplicaciones a JBoss, aunque de hecho es implementado por cualquier servidor de aplicaciones aun siendo este comercial. Esto le da a Compiere la flexibilidad en cuanto a plataforma de servidor de aplicaciones se refiere, el sistema operativo que es usado dependería del sistema operativo compatible con el servidor de aplicaciones que se vaya a usar, el propósito de esta investigación es proponer su uso sobre Linux, usando JBoss como servidor de aplicaciones (sobre esta combinación de tecnología fue desarrollado y probado al 100% por sus creadores).

Al hablar de un servidor de aplicaciones (en este caso JBoss), este envuelve varios módulos (contenedores), los contenedores y servicios que hace uso en los cuales Compiere fue desarrollado son los siguientes:

Compiere Server Services usando EAR (Enterprise Archive)

- Servidor de cuentas
- Servidor de peticiones
- Servidor de utilerías

Apache Web Server es un servidor Web que con un módulo de comunicación con el contenedor Servlet Tomcat se encargan de atender la petición del cliente Web. Tomcat tiene la capacidad de fungir como Web Server y contenedor de Servlets, en esta propuesta se decidió separar la funcionalidad debido a que el servidor Apache Web Server ha sido probado y este ha demostrado ser más eficiente (en cuanto a tiempo de respuesta) de contenido estático (HTML's), otra de la razones por la cual se propone usarlos separados es de que el Apache Web Server permite la interacción con interfaces como PHP (lenguaje de programación web) lo cual permite hacer páginas estáticas con un toque de contenido dinámico.

Al integrar la funcionalidad del Apache Web Server permite poder adicionar el uso de un sitio informativo hecho sólo con HTML, con el objetivo de atender las peticiones y preparar la respuesta a el cliente, de esta forma sería el Apache Web Server quien resuelva esa petición sin requerir de recursos del contenedor Tomcat, este escenario permitiría un mejor desempeño al canalizar los recursos correspondientes de manera eficiente (esto es solo cuando sean requeridos).

El contenedor de EJB's se usaría propiamente JBoss ya que esta funcionalidad solamente está proveída por JBoss.

El Sistema Operativo es Linux, siendo esta la plataforma más compatible para trabajar con Compiere ya que este fue desarrollado sobre las herramientas mencionadas en los párrafos anteriores.

Sobre Linux es usado un firewall como IPtables (revisado en el capítulo 3) esto asegura una protección contra ataques externos de clientes malintencionados.

Compiere hace uso de una base de datos de nivel empresarial para almacenar propiamente los datos procesados por el sistema, Compiere asegura el uso de una base de datos comercial, en el capitulo anterior se reviso el uso de MaxDB. Esta base de datos tiene la tecnología y robustez (tiempos de respuesta muy aceptables, tolerancia a fallas, administración) comparada con las bases de datos relacionales empresariales del mercado que actualmente se comercializan. MaxDB brinda el soporte necesario de todo lo que Compiere necesita para poder funcionar correctamente.

Esta arquitectura permite manejar Compiere vía una interfaz Web y preguntar su estatus de forma local o remota. Compiere tiene la capacidad de poder integrar extensiones de funcionalidad o alguna otra aplicación J2EE sin problemas.

4.3.3 Arquitectura Web

EL primer paso del hospedaje de aplicaciones es la caracteristica de una aplicación para ser desarrollada sobre Internet.

Muchas aplicaciones fueron diseñadas para la instalación de un cliente, lo que significa que el instalar otro cliente se necesitaría el mismo esfuerzo requerido para el cliente no. 1. Esto no es escalable y por lo mismo no deseable.

Para ser posible acceder desde múltiples clientes eficientemente en una sola base de datos y compartir datos. Compiere está diseñado para ser hospedado, lo que permite el poder accederlo desde cualquier parte, la información se mantiene dividida dependiendo de a quien pertenezca, conservando perfiles, configuraciones, datos, etc. Con una sola base de datos si es necesario.

Compiere hace fácil el agregar servicios, como operaciones (llamar a soporte, respaldo, etc.) y migración (nuevas versiones, interfaces, etc.). Además permite agregar servicios de soporte integrado para realizar tareas como mantenimiento de los catálogos del producto o servicios de cuenta y compra centrales.

En contraste con otras aplicaciones existentes (un ejemplo de una de ellas es SAP), Compiere tiene una estructura de información avanzada permitiendo cambios estructurales, permitiendo un manejo óptimo de cuentas y reglas.

4.3.4 Cambios Estructurales

Después de que una aplicación está en ambiente de producción o incluso durante la implementación, los usuarios constantemente requieren cambios en la estructura de información, existen muchas razones para llevar a cabo estas nuevas peticiones. Durante el uso de la aplicación, los usuarios se dan cuenta de que información no es necesaria o que información es requerida para la toma de decisiones, cambios en el negocio requieren la colección de información adicional.

Muchas aplicaciones no permiten ningún cambio, o un cambio de requerimientos requiere el mismo esfuerzo que si se implementara en caso de falla. Compiere permite a los usuarios (no al precisamente al personal técnico) agregar, cambiar o borrar dimensiones de información en cualquier momento. La estructura OLAP se mantiene de forma automática.

4.3.5 Dimensiones de Información

Compiere provee una extensa lista de dimensiones predefinidas, están son:

*Organización
*Cuentas Naturales
*Tiempo y Fecha
*Producto
*Categoría de producto
*Asociado de negocios (Business Partner)
*Proyecto
*Mercadotecnia

*Localidad (asociado de negocios, almacén)
*Actividad (actividad basada en costo)

Compiere permite a los usuarios definir dimensiones adicionales. Estas dimensiones son validadas vía listas o tablas. Todas las dimensiones de información permiten definiciones en diagrama de árbol, estos niveles resumidos permiten reflejar la estructura de organización, las hojas de balance o estados de ingreso, de igual manera los cambios dentro de la estructura de árbol, se hacen en cualquier momento. (Ver figura F4.1)

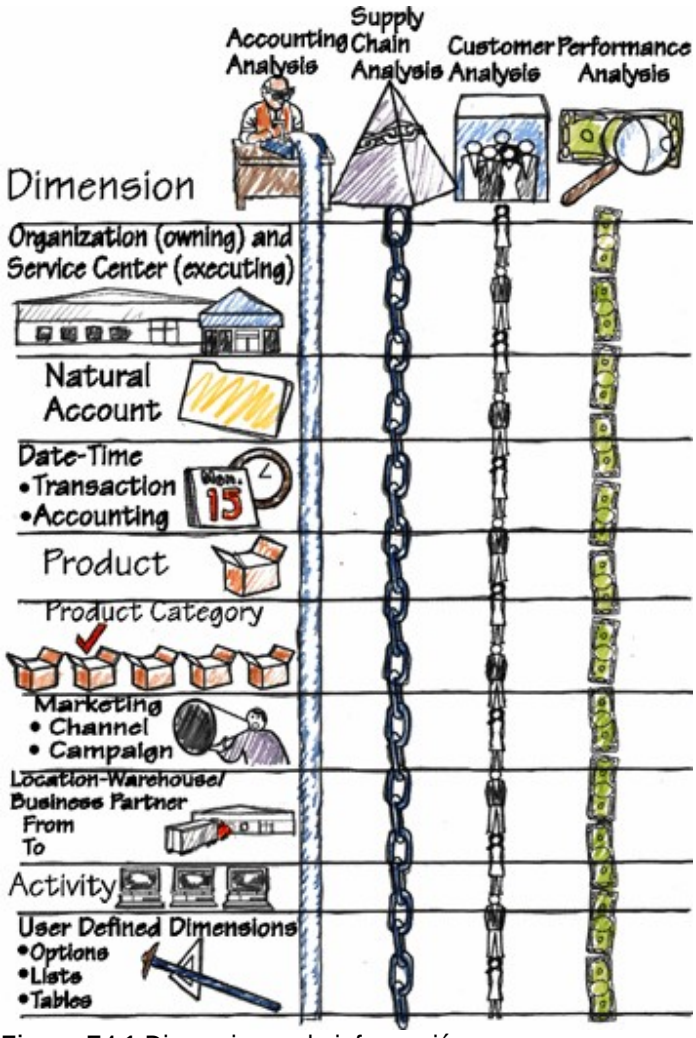

Figura F4.1 Dimensiones de información

Cada dimensión de información tiene un árbol primario y se tiene la capacidad de agregar más árboles, esto es requerido si se quiere mantener la vieja y nueva estructura de comparación o si se necesita diferentes jerarquías en paralelo.

4.3.6 Colección de datos automatizados

La entrada de datos necesita ser muy eficiente, lo que es equivalente a llenar el mínimo necesario de campos posibles. Compiere provee una interfaz, la cual facilita información automáticamente desde el contexto de transacción, los usuarios tienen la capacidad incluso de seleccionar opciones alternas.

Esta es una ventaja significativa, en donde usualmente la funcionalidad A no está "enterada" de la información que necesita la funcionalidad B, esto resulta en información que no está disponible o que no necesita estar derivada de datos existentes con la subsecuente pérdida de detalle.

Compiere está diseñado bajo un esquema multi-función, se pueden habilitar o deshabilitar en cualquier momento y cuando sea necesario.

Diseñando una aplicación con multi requerimientos en mente, la aplicación resultante es fácil de mantener y entender, e incluso la aplicación es más estable. Los beneficios medidos son de bajo costo de implementación y mantenimiento, adicionando una funcionalidad mucho más significativa.

4.3.7 Multi-Concurrencia

Las características de multi-concurrencia son:

Transacciones multi-concurrentes

- Habilidad para realizar transacciones con diferente moneda
- Habilidad para revalorar transacciones
- Cuentas bancarias en otro tipo de moneda

Reportes multi-concurrentes

- Habilidad para traducir transacciones o balances para propósitos de reporte

Cuentas multi-concurrentes

- Habilidad para realizar transacciones entre cuentas en forma paralela

4.3.8 Integración Funcional

Compiere integra totalmente la funcionalidad del recurso empresarial con la funcionalidad de relación con el cliente y proceso analítico, esta integración asegura que las diferentes áreas funcionales tienen toda la información requerida para la toma de decisiones del negocio.

4.3.9 Vistas de negocio

Si los reportes internos no son suficientes, las herramientas SQL de terceros tienen la capacidad de ser usadas en este punto. Compiere provee de vistas de negocio, que resuelven todas las referencias a llaves foráneas y están listas para usarse. No es necesario tener conocimiento del modelo de datos o de desarrollo y mantenimiento de catálogos por estas herramientas terceras

4.3.10 Exportación de Datos

Compiere exporta todos los datos en reportes usando los siguientes formatos:

- Microsoft Excel
- HTML
- XML
- Text
- PDF
- PS
- Microsoft Word
- Business OLAP Cubes

4.3.11 Importación de Datos

Compiere importa datos desde XML, registros arreglados, etc. Existen formatos predefinidos pero también se puede definir uno propio. Esto es importante debido a que en muchas ocasiones se requiere cargar el sistema con algún catalogo que se encuentra disponible en algún archivo.

4.3.12 Scanner, dispositivos electrónicos

Compiere provee una interfaz soporte de scanner para entrada de órdenes y transacciones de inventario como órdenes de compra, recepción de material.

4.3.13 Extensiones

Compiere provee la habilidad de extender la aplicación utilizando Java Business API s.

4.3.14 Scripting

El modo scripting (modo comandos) permite a los usuarios extender la funcionalidad usando sintaxis java. Esta modalidad incluso es usada para importar datos.

4.3.15 Integración de correo electrónico

Correos electrónicos multilingües son generados por el sistema para notificaciones como aprobaciones de requerimientos o alertas.

Los correos electrónicos incluso son usados para crear cartas personalizadas o confirmaciones automáticas.

Si un error ocurre, el usuario puede mandar el mensaje de error con la información de soporte generado por el sistema a una persona del soporte interno o incluso externo.

4.3.16 Ayuda

Compiere provee un integrado y personalizable sistema de ayuda multi-nivel. Cada tarea, reporte, ventana y pestaña tiene una explicación de información, cada campo tiene una recomendación "tip" con un texto de ayuda.

Si la ayuda no es suficiente, el usuario tiene la capacidad de saltar de la ventana de ayuda actual a la red con información actualizada relacionada al tema.

4.3.17 Procesos de Negocio

4.3.17.1 Manejo de Procesos de Negocios en Compiere

Entendamos por "Workflow" (flujo de trabajo) como los pasos que envuelve a la gente para realizar cierta tarea, el manejo de Procesos de negocio como workflow y actividades del sistema.

Compiere soporta totalmente el manejo de procesos de negocio (BPM) y está basado en estándares OMG. De aquí en adelante usaré el término de Workflow para incluir las capacidades de BPM.

Compiere comparado con otras aplicaciones de CRM y ERP, el workflow no se encuentra como la característica mas importante de la aplicación, pero para Compiere si, pues Compiere esta basado sobre workflow. El motor del Workflow es el propio núcleo de manejo de transacciones de Compiere. Esto significa que todos los procesos de workflow en Compiere son automáticamente habilitados, extendibles y modificables. Como workflow está totalmente integrado. Los workflow's

de Compiere son flexibles al mantenimiento y proveen mucha más funcionalidad que la externa o workflow agregados ofrecidos por otros ERP. Ver la figura F4.2.

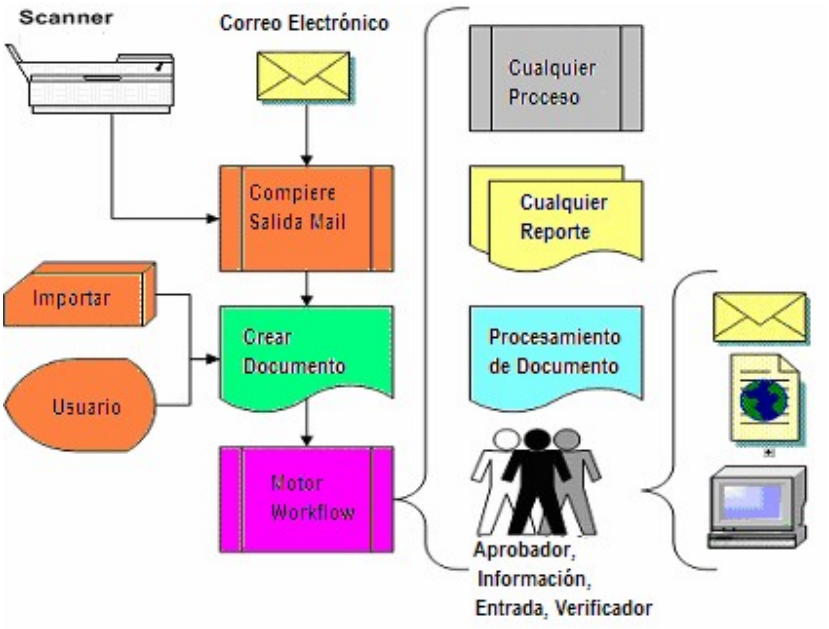

Figura F4.2. Workflow de ejemplo.

4.3.18 Tipos de Workflow

Compiere provee 3 tipos de workflow:

- Workflow general.- Provee una guía con instrucciones paso a paso para lograr una tarea. Ejemplos: los setup gráficos (Wizards) o procedimiento de fin de mes. El usuario los inicializa desde el menú
- Workflow de proceso de documento.- iniciado cuando procesas cualquier documento. Se podría extender este workflow para situaciones de aprobaciones, ejemplos: aprobación especial para órdenes sobre cierto capital.
- Workflow de valor de documento.- El workflow esta automáticamente inicializado cuando cualquier entidad satisface alguna condición definida por el usuario.

4.3.19 Nodos, Acciones y Transiciones

Un Compiere nodo Workflow (paso) puede tener las siguientes acciones:

- Proceso Automático.- cualquier proceso, reporte, tarea, workflow, acción, Web services, etc.
- Acción del usuario.- cualquier ventana, forma.
- Fijar variable.- cualquier columna a constante o variable.
- Opción del usuario.- cualquier opción (ejemplo: aprobación), listas de selección
- Espera(dormir).- puede ser usado para inicializar , terminar, etc.

4.3.20 Prioridad, Escalamiento, Alertas

Compiere provee un manejo de prioridades dinámicas, permitiendo el uso del motor workflow para llamar al centro de ruteo y prioridades basado en el soporte al cliente.

Se puede también definir reglas de escalamiento para deshabilitar y mandar alertas al workflow y/o supervisor. Aplicaciones hospedadas permiten a los clientes concentrarse en sus negocios.

4.3.21 Descripción de procesos de negocio

4.3.21.1 Cuotas

Se crean e imprimen cuotas de clientes basadas sobre precios de lista. Se tiene la capacidad de cambiar una cuota en cualquier momento y convertirla después en una orden de venta.

4.3.21.2 Orden de Ventas

Las órdenes de ventas son el "documento de control de cumplimiento". Las ordenes de ventas generan embarques y facturas y se cerciora que la orden se cumpla.

4.3.21.3 Embarques

Basada sobre la orden de venta, se generan uno o más embarques inmediatamente o automáticamente cuando en inventario este disponible.

4.3.21.4 Facturas de clientes

Facturas son generadas inmediatamente basadas sobre la orden de venta o basada sobre el embarque. Se puede escoger para crear una factura inmediatamente después de cada embarque, cuando la orden esta completamente embarcada sobre el calendario de facturación del cliente. Un calendario de facturación permite mandar resúmenes de facturas, ejemplo los embarques de fin de semana, mes, etc. Incluso se pueden ingresar manualmente las facturas.

4.3.21.5 Recibos

Cuando se ingresa una orden o factura, las regla de pago permiten generar automáticamente recibos

4.3.21.6 Estados Bancarios

Los estados de cuentas bancarios son ingresados manualmente o automáticamente cargados, ya sea creando pagos, ingresando cambios, etc.

4.3.21.7 Requisiciones

Se crean requisiciones manualmente o automáticamente. Se aprueban requisiciones para generar órdenes.

4.3.21.8 Ordenes de Compra

Genera y consolida órdenes de compra de requisiciones anteriormente aprobadas. La adquisición de material puede incluso directamente crear órdenes de compra. Se puede transmitir las órdenes de compra vía correo electrónico o fax.

4.3.21.9 Recepción de Material

Compiere tiene la capacidad de recibir material y directamente crear el cargamento directamente de la orden de compra(o factura del cliente).

4.3.21.10 Facturas del cliente

Directamente crea una factura desde la orden de compra o recibo. Incluso es capas de crear recibos automáticamente desde la factura cuando se recibe junto con el cargamento al mismo tiempo.

4.3.21.11 Pagos

Genera pagos basados en los términos de pago y de las ventajas de descuentos definidos. Se pueden hacer pagos a través de tarjeta de crédito, incluso salva esa información cuando se ingresa la orden de compra o factura.

4.3.21.12 Monitoreo de actividades

Compiere soporta lo siguientes tipos de peticiones

- Información- peticiones estructuradas y no estructuradas del web o correo electrónico
- Servicio.- petición estructurada para realizar un servicio en determinado momento y lugar
- Carga.- petición estructurada para rembolsar costos
- Cuenta.- petición estructura concerniente a un cliente en particular u orden de venta, cargamento, factura o pago
- Garantía.- petición estructurada concerniente a un asunto de un producto o servicio
- Ayuda.- petición estructurada de un servicio de un cliente

Dependiendo del tipo, la petición es automáticamente convertida determinado documento (orden, factura, oferta). Una confirmación por correo electrónico es enviada manualmente o automáticamente. Peticiones son asignadas a unos usuarios en particular.

Con el número de seguimiento, el creador puede actualizar la información, asegurando facilidades de administración en tiempos de respuesta.

Las peticiones incluso son generadas basadas sobre status de cuentas (fecha de la última venta, pago sobregirado, etc.) para servicio a clientes o ventas a seguir.

4.3.21.13 Administración de Campaña de Marketing

La conservación del cliente es una misión crítica para cada compañía. Compiere soporta esto creando correos o peticiones a la fuerza de ventas para dar seguimiento, ya que un mal criterio para una campaña puede ser la última venta. Se pueden medir volumen de venta aproximadas de acuerdo a la campaña de marketing que se esté manejando, también se puede saber que productos fueron comprados.

La efectividad de las campañas de marketing puede ser medida por el ingreso generado. Con una buena administración de una campaña se puede hacer un análisis de capacidad de ganancia.

Se reportan sobre la ganancia y el monto de ingreso de clientes en específico o grupos de clientes sobre un periodo de tiempo.

5 Conclusiones

El software libre ha permitido poder crear alternativas de solución para las empresas, en donde el gasto económico para tecnología debiera ser solamente de la infraestructura física (servidores e insumos necesarios para su funcionamiento) y personal intelectual que implemente esta solución. El tener acceso al código fuente tanto de la aplicación Compiere como de las herramientas de software permite tener un total control sobre la operación tecnológica y del propio negocio.

El ahorro de recursos financieros que pudieran provenir de gastos de licencias, mantenimiento y contratos de servicios de soporte técnico se reflejarían en la mejora de la operación diaria de una empresa al tener una mejor infraestructura tecnológica (hardware), uso de comunicaciones de vanguardia y presupuesto destinado para la contratación de personal con habilidades técnicas muy desarrolladas.

A lo largo del capítulo 4, se explicó la funcionalidad más importante que posee Compiere así como sus características siendo 100 % compatible con la propuesta de arquitectura de software libre siguiendo el estándar J2EE.

Compiere es uno de los muchos software libres que circulan por la Internet, atendiendo necesidades actuales de las empresas, resolviéndolos satisfactoriamente, convirtiéndose así en una alternativa viable económica, técnica y funcional.

La arquitectura tecnológica de Compiere permite incluso extender su funcionalidad, o comunicarlo con algún otro sistema propietario existente o por desarrollar. En materia legal, es totalmente libre de poderlo modificar sin ningún problema de violación de derechos de autor, siguiendo la filosofía explicada en el capítulo 1 referente a la libertad de modificación y uso.

Las características de la arquitectura en n-capas para aplicaciones empresariales de misión crítica, propuesta en el capítulo 3, permite extender sus capacidades con la posibilidad de integrar otros sistemas de la misma empresa o fuera de ella (explotando las capacidades que ofrece una arquitectura J2EE).

La arquitectura presentada permite la división lógica y funcional (capa de lógica de negocios y capa de presentación), teniendo así un completo control sobre las actualizaciones o nuevas implementaciones que se hagan, esto es, si se desea modificar la imagen visual presentada al cliente, se modifican los archivos de presentación (html) que se encuentran en el Apache Web Server sin correr el riesgo de cambiar la funcionalidad en el servidor de aplicaciones. Al mismo tiempo, si se requiere cambiar la funcionalidad directamente, se detectan los objetos(Servlets y JSP para el caso del Tomcat Container o de EJB's del EJB JBoss Container) involucrados, se crea el procedimiento de desplegado (deploy) de los nuevos objetos previamente probados sin necesidad de interrumpir el servicio que realiza el servidor.

Esta solución tuvo como objetivo el permitir brindar un servicio vital para el funcionamiento correcto de una empresa. Las herramientas (Servidores Web, aplicaciones, y datos) y Compiere

son suficientemente capaces de mantener una operación crítica para el negocio, ya que permite expandir, monitorear o actualizar su funcionalidad en cualquier capa.

Con esta investigación se demostró que con el uso del software libre, tal y como Compiere lo ofrece y siguiendo el estándar J2EE en la construcción de aplicaciones bajo una arquitectura de funcionamiento en capas, se construyen soluciones de calidad las cuales representan un servicio o servicios de misión críticos o de alta disponibilidad para una empresa.

Como estudiante de MAC (Matemáticas aplicadas y computación) fui capaz de desarrollar habilidades en el diseño de aplicaciones y conforme adquirí experiencia (una vez egresado) en el tema, me fui dando cuenta que es necesario crear un modelo de desarrollo estándar orientado a las empresas, ya que son éstas las que constantemente tienen que innovar tecnología y procedimientos para hacerlos más eficientes.

6 Glosario

GNU

El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completo libre: el sistema GNU . El 27 de septiembre de 1983 se anunció públicamente el proyecto por primera vez en el grupo de noticias net.unix-wizards. Al anuncio original, siguieron otros ensayos escritos por Richard Stallman como el "Manifiesto GNU", que establecieron sus motivaciones para realizar el proyecto GNU, entre las que destaca "retornar al espíritu de cooperación que prevaleció en los tiempos iniciales de la comunidad de usuarios de computadoras". GNU es un acrónimo recursivo que significa "GNU No es Unix". Stallman sugiere que se pronuncie, en inglés, como "guh-noo" (se puede observar que el logo es un ñu) para evitar confusión con "new" (nuevo). En español, GNU se pronuncia fonéticamente.

KDE

KDE (K Desktop Environment) es un entorno de escritorio gráfico e infraestructura de desarrollo para sistemas Unix y, en particular, Linux. La 'K', originariamente, representaba la palabra "Kool", pero su significado fue abandonado más tarde. Actualmente significa simplemente 'K', la letra inmediatamente anterior a la 'L' (inicial de Linux) en el alfabeto. Actualmente KDE es distribuido junto a muchas distribuciones Linux.

KDE imitó a CDE (Common Desktop Environment) en sus inicios. CDE es un entorno de escritorio utilizado por varios Unix.

De acuerdo con su página web, "KDE es un entorno gráfico contemporáneo para estaciones de trabajo Unix. KDE llena la necesidad de un escritorio amigable para estaciones de trabajo Unix, similar a los escritorios de MacOS o Windows".

GNOME

El proyecto GNOME (GNU Network Object Model Environment) surge en agosto de 1997 como proyecto liderado por los mexicanos Miguel de Icaza y Federico Mena para crear un entorno de escritorio completamente libre para sistemas operativos libres, en especial para GNU/Linux. Desde el principio, el objetivo principal de GNOME ha sido proporcionar un conjunto de aplicaciones amigables y un escritorio fácil de utilizar

API

Una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es un conjunto de especificaciones de comunicación entre componentes software.

Representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las APIs asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API.

Arpanet

La red de computadoras ARPANET fue creada por encargo del Departamento de Defensa de los Estados Unidos como medio de comunicación para los diferentes organismos del país. El primer nodo se creó en la Universidad de California y fue la espina dorsal de Internet hasta 1990, tras finalizar la transición al protocolo TCP/IP en 1983.

UNIX

Unix® (o UNIX) es un sistema operativo portable, multitarea y multiusuario; desarrollado en principio por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Hoy día, la palabra UNIX se utiliza para denotar diferentes conceptos dependiendo del contexto en que es usada. Esto suele dar lugar a confusiones:

- UNIX - familia: desde el punto de vista técnico, UNIX se refiere a una familia de sistemas operativos que comparten unos criterios de diseño e interoperabilidad en común. Esta familia incluye más de 100 sistemas operativos desarrollados a lo largo de 20 años. No obstante, es importante señalar que esta definición no implica necesariamente que dichos sistemas operativos compartan código o cualquier propiedad intelectual.
- UNIX - el sistema operativo original: desde el punto de vista histórico, UNIX se refiere a la subfamilia de sistemas operativos que descienden de la primera implementación original de AT&T. El término "descendencia" ha de interpretarse como trabajos derivativos que comparten propiedad intelectual con la implementación original.
- UNIX - la marca: desde el punto de vista legal, Unix es una marca de mercado. Dicha marca es propiedad de "The Open Group," una organización de estandarización que permite el uso de dicha marca a cualquier sistema operativo que cumpla con sus estándares Públicados. Todo ello independientemente de que el sistema operativo en cuestión sea descendiente o clónico del Unix original. Resumiendo, la marca Unix no es propiedad de ninguna compañía.

.

Porting

la acción de traducir software para correr sobre una diferente computadora y/o sistema operativo

Firewall

Un cortafuegos (o firewall en inglés), es un elemento de hardware o software utilizado en una red de computadoras para prevenir algunos tipos de comunicaciones prohibidos según las políticas de red que se hayan definido en función de las necesidades de la organización responsable de la red. Su modo de funcionar es definido por la recomendación RFC 2979, la cual define las características de comportamiento y requerimientos de interoperabilidad.

La idea principal de un cortafuegos es crear un punto de control de la entrada y salida de tráfico de una red. Un cortafuegos correctamente configurado es un sistema adecuado para añadir protección a una instalación informática, pero en ningún caso debe considerarse como suficiente. La Seguridad informática abarca más ámbitos y más niveles de trabajo y protección.

B2B

Abreviatura comercial de la expresión anglosajona business to business: comercio electrónico entre empresas.

El comercio electrónico es una utilidad más que aporta Internet y que ha experimentado un gran auge en los últimos años. El B2B ha venido impulsado fundamentalmente por la creación de portales para agrupar compradores. Así, encontramos, por ejemplo portales de empresas de automoción, alimentación, químicas u hostelería, entre otros. Las compañías se agrupan para crear dichas páginas aglutinando fuerzas lo que les permite negociar en mejores condiciones. El mantenimiento de las páginas se produce pidiendo un canon por cotizar o cobrando a los socios una comisión del negocio realizado en el portal.

Algunas de las ventajas que aporta el B2B para los compradores son:

- Posibilidad de recibir mayor número de ofertas.
- Despersonalización de la compra con lo que se evitan posibles tratos de favor.
- Abaratamiento del proceso: menos visitas comerciales, proceso de negociación más rápido, etc. Por tanto, los compradores pueden pedir una reducción de precios en virtud del menor coste de gestión.

B2C

B2C es la abreviatura de la expresión business to consumer (empresas a consumidor), es decir, el comercio electrónico que realizan las empresas con los particulares. Potencialmente, tiene un gran recorrido a largo plazo y en la actualidad se va asentando en sectores como la distribución alimentaria. Así, las grandes cadenas de supermercados e hipermercados ya disponen en sus portales de aplicaciones de venta a través de internet. Otro ejemplo en B2C es el mayorista estadounidense de libros, música y otros productos Amazon.com.

El éxito del B2C pasa por el aseguramiento de los sistemas de pago a través de tarjeta de crédito, si bien en muchos casos se da la posibilidad de otras formas de pago como contra reembolso, en efectivo o la utilización de servicios proporcionados por otras empresas como PayPal.

E-procurement

E-Procurement es una expresión anglosajona que designa la versión cibernética de una cooperativa. En esta modalidad de comercio electrónico, las empresas se unen para comprar a través de internet algunos servicios y productos no estratégicos que necesitan para su actividad. Por ejemplo, el mobiliario o material de oficina: sillas, bolígrafos, papel, etc.

Al agrupar su demanda las compañías obtienen un mayor poder de negociación pudiendo presionar sobre los precios de compra. Con el uso de internet, además, consiguen canalizar mejor las diferentes ofertas.

En el Ecuador el término e-procurement es usado generalmente para las operaciones que realiza el gobierno electrónico entre las empresas del Estado y los usuarios del Estado, es decir con la implementación de la Ley de Comercio Electrónico y Firmas Electrónicas cada ecuatoriano puede hacer, casi cualquier trámite por Internet y tener la constancia de que se produjo ese trámite.

E-commerce

El Comercio Electrónico, E-Commerce, Electronic Commerce, EC, e-commerce ó ecommerce consiste principalmente en la distribución, compra, venta, marketing y suministro de información complementaria para productos o servicios a través de redes informáticas como Internet u otras redes informáticas. La industria de la tecnología de la información podría verlo como una aplicación informática dirigida a realizar transacciones comerciales.

Una definición alternativa la vería como la conducción de comunicaciones de negocios comerciales y su dirección a través de métodos electrónicos como intercambio electrónico de datos y sistemas automáticos de recolección de datos.

El comercio electrónico también incluye la transferencia de información entre empresas

E-business

El hacer Negocios Electrónicos (e-Business) integra no solo el E-Commerce sino también la operativa interna, por ende se accesa a la infraestructura informática, los procesos de las ventas electrónicas, en definitiva toda la administración de un negocio que está conectado a una página web y las transacciones que en ella se desencadenen. En términos realmente simples podemos decir que cuando alguien realiza una compra en un sitio web, esa transacción se refleja de manera

inmediata en los sistemas informáticos de la empresa, a su vez que dispara los procesos administrativos, financieros y de despacho necesarios.

GIF

GIF (Graphics Interchange Format) es un formato gráfico utilizado ampliamente en la World Wide Web, tanto para imágenes como para animaciones.

El formato fue creado por CompuServe en 1987 para dotar de un formato de imagen a color para sus áreas de descarga de ficheros, sustituyendo su temprano formato RLE en blanco y negro. GIF llegó a ser muy popular porque podía usar el algoritmo de compresión LZW (Lempel Ziv Welch) para realizar la compresión de la imagen, que era más eficiente que el algoritmo Run-Lenght Encoding (RLE) que usaban formatos como PCX y MacPaint. Por lo tanto, imágenes de gran tamaño podían ser descargadas en un razonable periodo de tiempo, incluso con modems muy lentos.

GIF es un formato sin pérdida de calidad, siempre que partamos de imágenes de 256 colores o menos. Una imagen de alta calidad, como una imagen de color verdadero (profundidad de color de 24 bits o superior) debería reducir literalmente el número de colores mostrados para adaptarla a este formato, y por lo tanto existiría una pérdida de calidad.

Persistencia

Se define por persistencia en el mundo de la Orientación a Objetos, como la capacidad que tienen los objetos de sobrevivir al proceso padre que los creo. Esto es, que su ciclo de vida excede de la del programa que lo instancia.

IIOP

En computación distribuida, GIOP (Protocolo Entre ORBs General, General Inter-ORB Protocol) es el protocolo abstracto por el cual los ORBs se comunican. Los estándares asociados con el protocolo son mantenidos por el Object Management Group (OMG).

IIOP (Internet Inter-Orb Protocol) es la impleentación deGIOP para TCP/IP. Es una realización concreta de las definiciones abstractas de GIOP.

SOAP/HTTP

SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

RPC (JAX-RPC)

El RPC (del inglés Remote Procedure Call, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo fue propuesto inicialmente por Sun Microsystems como un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro del paradigma cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación al cliente.

Hay distintos tipos de RPC, muchos de ellos estandarizados como pueden ser el RPC de Sun (RFC 1057), Distributed Computing Environment (DCE), DCOM de Microsoft. No siendo compatibles entre sí. La mayoría de ellos utilizan un lenguaje de descripción de interfaz (IDL) que define los métodos exportados por el servidor.

RSS

RSS es parte de la familia de los formatos XML desarrollado específicamente para todo tipo de sitios que se actualicen con frecuencia y por medio del cual se puede compartir la información y usarla en otros sitios web o programas. A esto se le conoce como redifusión o sindicación.

El RSS no es otra cosa que un sencillo formato de datos que es utilizado para sindicar (agregar, suscribir) contenidos a suscriptores de un sitio web. El formato permite distribuir contenido sin necesidad de un navegador, lo cual también puede verse como desventaja ya que necesita de la instalación de otro software. Algunos adelantos han permitido utilizar el mismo navegador para ver los contenidos RSS mediante programación de los denominados scripts de interpretación. Así también las nuevas versiones de los navegadores permitirán leer los RSS sin necesidad de software adicional, por el momento son versiones betas

mod_jk2

Conector que permite la comunicacion entre el Apache Web ser y Tomcat Servlet container.

SEI

Software Engineering Institute (SEI) es un instituto federal americano de investigación y desarrollo,fundado por el Congreso Americano en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software, que dieran respuesta a los problemas que generaba al ejército americano la programación e integración de los sub-sistemas de software en la

construcción de complejos sistemas militares. Financiado por el Departamento de Defensa americano y administrado por la Universidad Carnegie Mellon.

Es un referente en Ingeniería de Software por realizar el desarrollo del modelo SW-CMM (1991) que ha sido el punto de arranque de todos los que han ido formando parte del modelo que ha desarrollado sobre el concepto de capacidad y madurez, hasta el actual CMMI.

SEI alberga también a otro instituto federal de investigación y desarrollo 3: CERT, fundado por SEI en noviembre de 1988 por encargo de DARPA (Defense Advanced Research Projects Agency) para investigar y mejorar la seguridad de los sistemas de información del ejército y ejercer la coordinación en caso de emergencias.

ASP

Un Proveedor de Servicios de Aplicaciones (PSA; en inglés application service provider, ASP) es una empresa que proporciona servicios de software a múltiples entidades desde un centro de cómputo a través de una red. Entre los factores que habilitan a un PSA se destacan la amplia difusión del uso de Internet, la capacidad de acelerar el despliegue y puesta en marcha de aplicaciones y la posibilidad de transferir servicios y operaciones a terceros. La barrera principal para un PSA radica en convencer a sus clientes de que su información en manos de un tercero permanece segura.FF

ISP

Un proveedor de servicios de Internet (o ISP por el acrónimo inglés de Internet Service Provider) es una empresa dedicada a conectar a Internet la línea telefónica de los usuarios o las distintas redes que tengan, y dar el mantenimiento necesario para que el acceso funcione correctamente. También ofrecen servicios relacionados, como alojamiento web o registro de dominios entre otros.

EAR

Ear (del ingles enterprise archive) es un archivo comprimido .jar que contiene una aplicación J2EE. Una aplicación es un grupo de módulos Web que colectivamente realizan una tarea en común.

7 Bibliografía

Libros

- La Catedral y el Bazar (pionero de la tematica Software Libre) (traducción por José Soto Perez). (1998). Autor: Erick S. Raymond.

- Redes (parte 3. Protocolos de red). Autor: Craig Zacker. McGraw-Hill, 2001

- Developing Java Applications with J2EE and UML (chapter 1. Introduction to Enterprise Software (evolution to enterprise software)). Autores: Khawar Zaman Ahmed,Cary E. Umrysh. Addison –Wesley, Dic 15 2001.

- J2EE –Manual de Referencia. (cap. 1. Vision General y cap. 2. Arquitectura Multicapas), Autor: Jim Keogh, Mc Graw Hill, 2002.

Sitios Internet

Juan Antonio Martinez (11 de Abril del 1999). "La empresa ante el software libre ".[en linea]. http://oasis.dit.upm.es/~jantonio/documentos/empresa/empresa.html#toc4 . [consulta marzo del 2004]

Free Software Foundation Inc. (22 de octubre del 2006, actualización). "El Software Libre ".[en linea]. http://www.gnu.org/philosophy/free-sw.es.html .[consulta 11 noviembre del 2006]

Free Software Foundation Inc.(2000-2007). [en linea] http://www.fsf.org/ .[consulta marzo del 2004]

Free Software Foundation Inc. (1996-2006). "Licences ".[en linea]. http://www.gnu.org/licenses/ .[consulta marzo del 2004]

www.hipatia.info (octubre del 2003). "Definiciones y conceptos relativos al Software Libre "(version pdf).[en linea]. http://docs.hipatia.info/dsl/definiciones.pdf [consulta marzo del 2004]

Free Software Foundation Inc. (1996-2006) "Categorías de software libre y no libre ".[en linea]. http://www.gnu.org/philosophy/categories.es.html. [consulta marzo del 2004]

Juan R. Pozo (2001-2003). "Breve historia de la World Wide Web ".[en linea]. http://html.conclase.net/articulos/historia [consulta 20 de agosto del 2005]

Dan Connolly. (2000) "A Little History of the World Wide Web ".[en linea]. http://www.w3.org/History.html [consulta 20 de agosto del 2005]

Jorge Espinosa. (13 de Enero del 2003). "Introducción a n-Capas con VFP y VB". [en linea]. http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art20.asp. [consulta 16 de agosto 2005].

Alejandro Benavides. (31 de enero del 2006). "Programacion en N Capas bajo Gambas". [en linea]. http://www.mygnet.com/articulos/gambas/381/ .[consulta 11 de septiembre del 2006]

www.javaHispano.org (Martin) (2001-2005). "¿Y Sabes cuáles son las características generales de un sistema de n-capas?". (Nivel conceptual desde un punto de vista lenguaje de programación). http://www.javahispano.org/tips.item.action?id=41. [consulta 13 de Abril del 2005]

apuntes.rincondelvago.com (Victor Israel - Mexico) . "Modelo OSI (Open Systems Interconnection) de la ISO (International Standard Organization) "(version pdf). http://apuntes.rincondelvago.com/modelo-osi-de-la-iso.html .[consulta 22 de Agosto del 2005].

Rhys Haden (1996 –2007). "The OSI Model". http://www.rhyshaden.com/osi.htm .[consulta 14 de enero del 2007].

www.protocols.com . "Protocols Directory" http://www.protocols.com/protocols.htm#physical. [consulta 22 de Agosto del 2005]

www.aunmas.com "Diferencias entre Internet, Intranet y Extranet". http://www.aunmas.com/guias/intranet/int_diferencias.htm. [consulta 3 septiembre del 2006]

www.nodo50.org, www.nodo50.net (aportaciones de la comunidad de software libre) (sin fecha) "Los dominios Internet". http://www.nodo50.org/manuales/internet/3.htm .[consulta 5 de noviembre del 2005].

es.wikipedia.org (12 de febrero del 2006). "Lenguaje Unificado de Modelado". [en linea] http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado . [consultado 23 de marzo del 2006]

Alexandra Rey Martinez, Marcela Leal Aponte, Carlos Andrés Gonzalez (2007). "UML = Unified Modeling Language ,Lenguaje Unificado de Modelamiento". [en linea].http://www.creangel.com/uml/home.php . [consultado 13 de mayo del 2006]

The JSR-151 Expert Group (Larry W. Allen (SilverStream Software), Karl Avedal (Individual), Charlton Barreto (Borland Software Corporation), Edward Cobb (BEA), Alan Davies (SeeBeyondTechnology Corporation), Sreeram Duvvuru (iPlanet), B.J. Fesq (Individual), Mark Field (Macromedia), Mark Hapner (Sun Microsystems, Inc.), Pierce Hickey (IONA), Hemant Khandelwal (Pramati Technologies), Jim Knutson (IBM), Elika S. Kohen (Individual), Ramesh Loganathan (Pramati Technologies), Jasen Minton (Oracle Corporation), Jeff Mischkinsky (Oracle Corporation), Richard Monson- Haefel (Individual), Sean Neville (Macromedia), Bill Shannon (Sun Microsystems, Inc.), Simon Tuffs (Lutris Technologies), Jeffrey Wang (Persistence Software, Inc.), and Ingo Zenz (SAP AG)) . (24 de noviembre del 2003) . "Java™2 Platform Enterprise Edition

Specification, v1.4 ". [en linea] . http://java.sun.com/javaee/reference/ [consultado 22 de agosto del 2005]

Evelyn Zapata, Dario Cutis (23 de mayo de 2006) "Estudio Revela Que Piratería de Software para PC Aumentó sos Puntos Porcentuales en Latinoamérica en el 2005 ".[en linea]. http://www.bsa.org/mexico/press/newsreleases/2006-Global-Piracy-Study.cfm [consultado 15 Julio del 2006]

Pablo Chamorro Constain (10 de junio de 1999). "El Software Libre ". [en linea]. http://www.geocities.com/sl_edu_colombia/soluciones/pablo/slibre3.htm . [consultado 14 de junio 2004]

Rafael Martinez (1998-2006). "El Rincón de Linux para Hispanohablantes ". [en linea]. http://www.linux-es.org/articulos (notas diversas). [consultado 18 de Abril del 2006].

Pello Xabier Altadill Izura . "IPTABLES manual practico, tutorial de iptables con ejemplos ". [en linea]. http://es.tldp.org/Manuales-LuCAS/doc-iptables-firewall/doc-iptables-firewall.pdf [consultado 20 de mayo del 2005]

Jesús Vegas (21 marzo del 2002). "El Navegador Web, Browser ". [en linea]. http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node19.html [consultado 22 mayo del 2005]

Felipe Aguilera. "Patrones de diseño ". [en linea]. http://www.dcc.uchile.cl/~luguerre/cc40b/clase13.html .[consultado 20 mayo del 2005]

Apache Software Foundation (1999 –2005) "Apache Web Server ". [en linea]. http://httpd.apache.org/ . [consultado 20 de Agosto de 2005].

UnoRed (12 de Abril del 2003). "Nuestra Infraestructura ". [en linea] . http://www.unored.com/modules.php?name=Content&pa=showpage&pid=23 .[consultado 25 de Agosto de 2005]

Red Hat (2007) "Capítulo 10. Servidor Apache http ". [en linea]. http://www.europe.redhat.com/documentation/rhl9/rhl-rg-es-9/ch-httpd.php3 .[consultado 10 enero 2007]

Raquel CEDAZO LEÓN (Julio 2005). "LABORATORIO FÍSICO REMOTO ACCESIBLE VÍA WEB PARA CONTROLAR UN BRAZO ROBOT INDUSTRIAL EN TIEMPO REAL ". (pagina 41). [en linea]. www.ciclope.info/docs/TFC-rcedazo.pdf [consultado 10 Octubre del 2005]

Cafesoft . "Tomcat Security Overview and Análisis ". [en linea] . http://www.cafesoft.com/products/cams/tomcat-security.html [consultado 4 de febrero del 2006]

Apache Software Foundation (1999-2006). "Apache Tomcat Overview ". [en linea]. http://tomcat.apache.org/tomcat-6.0-doc/config/index.html .[consultado 20 de Agosto 2005]

JBoss Inc. Group (2006). "JBoss Application Server ". (version pdf) [en linea]. http://www.jboss.org/pdf/JBossASBrochure-Mar2006.pdf. [consultado 10 de Enero del 2007].

MySQL AB (1995-2007) "MaxDB™ ". (Sección Documentation) [en linea] . http://www.mysql.com/products/maxdb/ [consultado 10 Enero del 2007].

Jose Camilo Deccach (13 de enero del 2006). "Internet de misión critica ". [en linea]. http://www.deltaasesores.com/prof/PRO147.html [consultado 20 de marzo del 2006].

TenStep Latinoamerica (PMI) "0.0.1.1 Modelo de Madurez de Capacidades (CMM) "[en linea]. http://www.tenstep.com.mx/Paso0.0.1.1.asp [consultado 16 septiembre del 2006]

CSCL. "CS/10,000 and the Capability Maturity Model ". [en linea]. http://www.cscl.com/techsupp/techdocs/cmmwp.html [consultado 16 de septiembre del 2006].

Jose Camilo Deccach "PRESENCIA EN INTERNET - MISIÓN CRÍTICA ". [en linea]. http://www.gestiopolis.com/delta/prof/PRO004.html. [consultado 22 de marzo del 2006].

El equipo de The Inquier (31 de Diciembre del 2006) "LINUX despega en los negocios de misión critica ". http://es.theinquirer.net/2006/12/31/linux_despega_en_los_negocios.html . [consultado 12 Enero del 2007].

ComPiere Inc. (1999-2006) "Compiere Documentation ". [en linea]. http://www.compiere.org/documentation/index.html. [consultado 22 de Marzo del 2006].

Emilio Carpio (10 de noviembre del 2004). "Existen muchas razones para usar Firefox ". [en linea]. http://www.aemilius.net/soporte/noticias/razones-para-usar-Firefox-sobre-otros-navegadores.html . [consultado 10 junio del 2006]

Mozilla foundation (29 de noviembre del 2006). "Firefox Caracteristicas ". [en linea] . http://www.mozilla-europe.org/es/products/firefox/features/#experience . [consultado 13 de dciembre del 2006]