UNAM POSGRADO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA ELÉCTRICA

(INFORMÁTICA)

DISEÑO DE APLICACIONES DISTRIBUIDAS, ARQUITECTURA Y METODOLOGÍA

PARA OPTAR POR EL GRADO DE

MAESTRO EN INGENIERÍA

PRESENTA

LUIS MANUEL TOVAR ESCOBAR

TUTOR: Dr. ABEL HERRERA CAMACHO

Año 2005

INDICE

INTRODUCCIÓN

Este trabajo presenta de una manera sencilla la gran complejidad a la que los desarrolladores de sistemas, arquitectos de sistemas, operadores y gerentes de sistemas tienen que enfrentarse al momento de diseñar y operar un sistema distribuido. Ahora ya no es suficiente entender el lenguaje de programación o técnica, ahora es necesario también romper el paradigma de sistemas que creó el enfoque cliente servidor, también es necesario vencer la resistencia al cambio de la mayoría de las instituciones y además vencer la rigidez de la operación de sistemas que hoy por hoy se basan en la orientación a sistemas y que hacen que la mayoría de las aplicaciones distribuidas no tengan éxito o sean muy costosas en su implementación, por ello el lector que desee poder entender toda la complejidad de una aplicación distribuida y que quiera conocer todos los aspectos necesarios para su implementación debe apoyarse en este trabajo ya que es una guía para el análisis, desarrollo, implementación y operación de este tipo de aplicaciones. El trabajo se enfocará a describir la forma en que una ***Enterprise Application (Aplicación Empresarial)*** puede ser arquitectada, diseñada y puesta en operación basándose tanto en el ***Modelo de Múltiples Capas*** como en el estándar **J2EE (Java 2 Enterprise Edition)**.

Se dará una breve explicación de lo que es J2EE y cómo ayuda en la creación de aplicaciones empresariales, también se propondrá una forma de dividir la funcionalidad en capas, se presentará un diseño lógico y físico en donde el lector podrá diferenciar las diferentes responsabilidades de cada capa y de cada componente de éstas. Se describirá la forma en que este tipo de aplicaciones debe ser construida, arquitectada y la forma en cómo los componentes pueden ser reutilizados, esto último como consecuencia de un buen diseño ya que cada componente puede (o al menos debería) servir para más de una aplicación. En otra parte del trabajo se describirá qué es y para qué sirve el servidor de aplicaciones, ya que esta pieza constituye la médula espinal de las aplicaciones distribuidas porque es en esta plataforma en donde se encuentra mayormente el corazón de los sistemas distribuidos al ser este componente el que puede proveer la capacidad para tolerancia a fallas, balanceo, reutilización, concentración de objetos y monitoreo de las aplicaciones.

La figura 1 representa la arquitectura típica de una aplicación empresarial distribuida, y este tipo de aplicaciones junto con su arquitectura es la que esté trabajo ayudará a construir tomando como base el estándar J2EE. Además de lo anterior, se propondrá una metodología para la construcción de estas aplicaciones, y se enunciarán todos los elementos necesarios para su exitosa implementación en ambientes de producción.

Los componentes mostrados en la figura 1 son:

Teléfono Celular	Es un tipo de cliente en donde se utilizá un minibrowser en el celular para acceder a la aplicación distribuida y hacer uso de ella.
Web Browser y otros medios de entrega	El cliente utilizará el Web Browser para acceder a la aplicación y hacer uso de ella, sin embargo existe la posibilidad de contar con algún otro medio de entrega o interfase con el usuario final.
Web Server	En esta capa se encontrarán todos aquellos elementos que tengan presentación estática, es decir, que no necesiten lógica de negocio o de procesamiento.
Servidor Aplicativo	En esta capa intermedia se encontrarán todos aquellos elementos que tengan presentación dinámica, es decir, que tengan lógica de procesamiento o lógica de negocio.
Base de Datos	En esta última capa se ubicarán los datos para que la aplicación pueda funcionar así como los datos de los clientes.

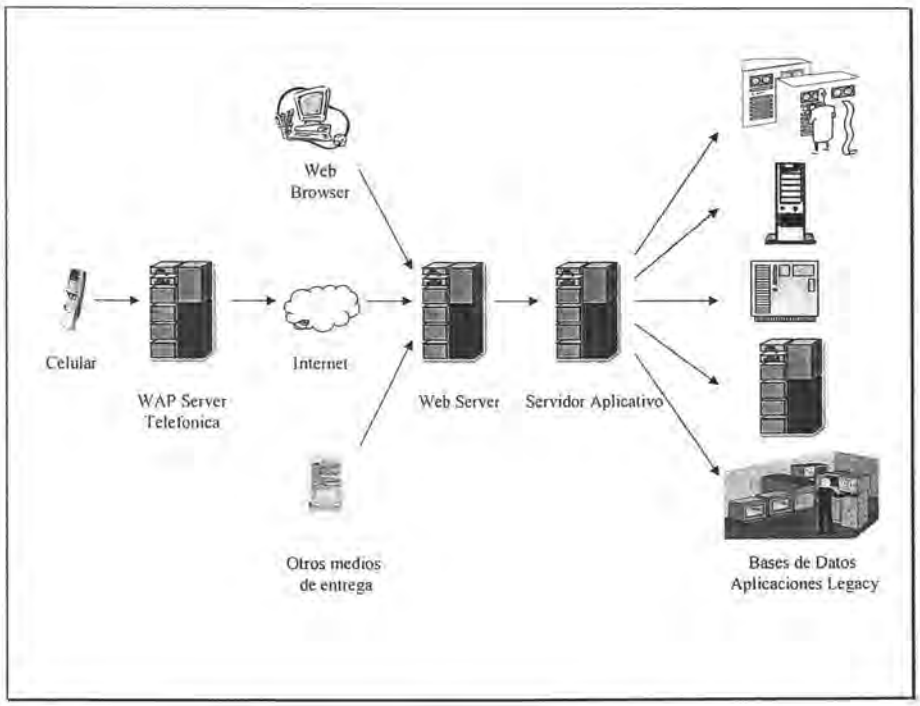

Figura 1 Componentes de una Aplicación Empresarial Internet / Intranet

Como puede observarse en la figura 1 y en la tabla de sus componentes, esta aplicación puede atender a diferentes tipos de clientes (Celular, Browser, Aplicación) ubicada en la WAN interna (Intranet) o en Internet. A lo largo del trabajo, el lector se podrá dar cuenta el porque el enfoque cliente servidor deja de ser funcional cuando se quiere atender a clientes que están dentro y fuera de la compañía, brindándoles un servicio 7 x 24, y es que el reto de dar un servicio ininterrumpido, distribuido, que de servicio en los horarios menos comunes o en las horas pico resulta inmenso.

De esta manera se comenzará en el capítulo 1 con una muy breve historia de cómo han evolucionado las aplicaciones distribuidas desde el modelo cliente servidor, se definirán este tipo de aplicaciones, se realizará una corta explicación de los principales subsistemas del estándar J2EE, se darán las características de una aplicación distribuida y se explicará cuál es el problema y complejidad al que los diseñadores y desarrolladores se enfrentan con este tipo de aplicaciones.

Ya en el segundo capítulo comienza la propuesta planteada en donde bajo el principio de "divide y vencerás" se presentan una serie de componentes lógicos separados e identificados en diferentes capas, el propósito de este capítulo es particionar a una aplicación distribuida en capas con diferentes responsabilidades. Dentro de cada capa se proponen una serie de componentes de software que tienen la propiedad de ser reusables y tienen una funcionalidad definida dentro de la capa en la que se encuentran.

En el tercer capítulo se presentará el método propuesto para diseñar aplicaciones distribuidas, se mostraran diagramas de la parte física que soportará a la lógica de la división en capas, igualmente se describe la manera en que los diferentes componentes físicos y lógicos interactúan dentro de un servidor aplicativo y un monitor de transacciones.

En el cuarto capítulo se habla de la forma en que se debe realizar la implementación, es decir, la construcción del sistema y todos los elementos que se deben considerar para construir eficazmente una aplicación distribuida. Este capítulo y el siguiente tocan aspectos mas de organización que técnicos, son los puntos que generalmente se olvidan pero que tienen un impacto determinante sobre el sistema.

Así, en el quinto capítulo se presentan los aspectos de pruebas y operación del sistema que muchas veces no son tomados con la importancia que tienen, se habla de los contratos de servicio que se tienen que firmar con las áreas responsables de la operación y por supuesto de la importancia de las pruebas de funcionalidad y de estrés, así como de la planeación de capacidades para que el sistema soporte la carga esperada.

Finalmente se presentan conclusiones.

OBJETIVO

El objetivo de este trabajo consiste en plantear todos los elementos que deben considerarse para la exitosa implementación de un sistema distribuido, desde su concepción, diseño arquitectónico fisico, el diseño arquitectónico lógico, es decir, la división en capas de la funcionalidad, las consideraciones de seguridad, de aseguramiento de la calidad, de la planeación de capacidades, de implementación, la post implementación, el monitoreo, los contratos de servicio y en general de todos los aspectos que involucren el diseño, puesta en marcha y operación de un sistema distribuido.

Al final de la lectura el lector podrá reconocer y entender los conceptos que influyen para la operación exitosa de un sistema distribuido, desde su concepción hasta su operación, sin embargo este documento no es una referencia a todos los temas tratados en él, es decir, más que una referencia de tecnologías, es una guía para la construcción de sistemas distribuidos ya que toma dichas tecnologías y las utiliza a favor de esta construcción.

El trabajo se enfoca en los puntos anteriormente señalados, sin embargo la parte de análisis y diseño de sistemas queda fuera del alcance ya que se considera que existe mucha literatura sobre dichos temas, este documento por el contrario pretende proporcionar una guía para el diseño de sistemas distribuidos, sacándole provecho al estándar J2EE, a los Servidores de Aplicaciones y a toda la nueva tecnología disponible para aprovechar el poder de computo de N cantidad de elementos de procesamiento.

CAPÍTULO 1

EL DESARROLLO DE APLICACIONES EMPRESARIALES

1.1 Antecedentes

Las primeras aplicaciones que utilizaron motores de base de datos fueron conocidas como Aplicaciones Cliente – Servidor, este tipo de aplicaciones consisten en un cliente en donde esta prácticamente toda la lógica de negocio que se comunica por algún medio a una base de datos en donde residen los datos necesarios para que la aplicación pueda funcionar. El esquema siguiente representa a este tipo de aplicaciones.

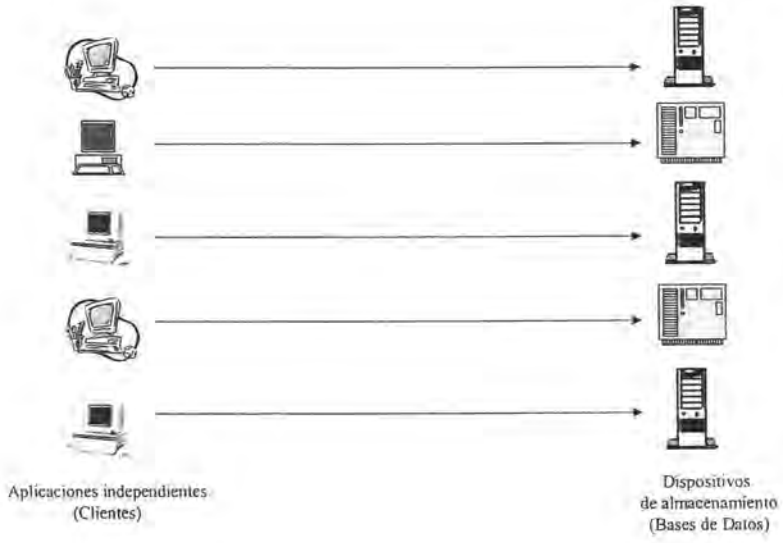

Figura 1.2 Modelo Cliente Servidor

Tecnología sobre la cual se implementaban estas soluciones:

Cliente	Servidor
Computadora Personal	HPUX Server E1000
Procesador Intel 386 o superior.	Procesador RISC.
100 Mhz	100 Mhz
128 MB en RAM	1 GB en RAM
Sistema Operativo: DOS o Windows	Sistema Operativo: Unix
Software aplicativo: Visual Basic 3	Base de Datos: Sybase 9
Cliente Base de Datos: DB Library	

Este esquema funcionó bien hasta que las aplicaciones dejaron de ser independientes, es decir, se vieron en la necesidad de interactuar unas con otras "en línea". Este tipo de relaciones no se concibió a tiempo y cuando fue necesario comunicar una aplicación con otra los ingenieros de software se vieron en la necesidad de realizar conexiones que a la larga resultan costosas y difíciles de administrar debido a que cada aplicación cliente debía implementar sus propios mecanismos de conexión hacia el servidor, esto creaba una verdadera telaraña de conexiones que repercutía en el tráfico en la red, además de que las aplicaciones estaban restringidas a un área geográfica reducida (el mismo edificio por ejemplo), la siguiente figura ilustra lo anterior.

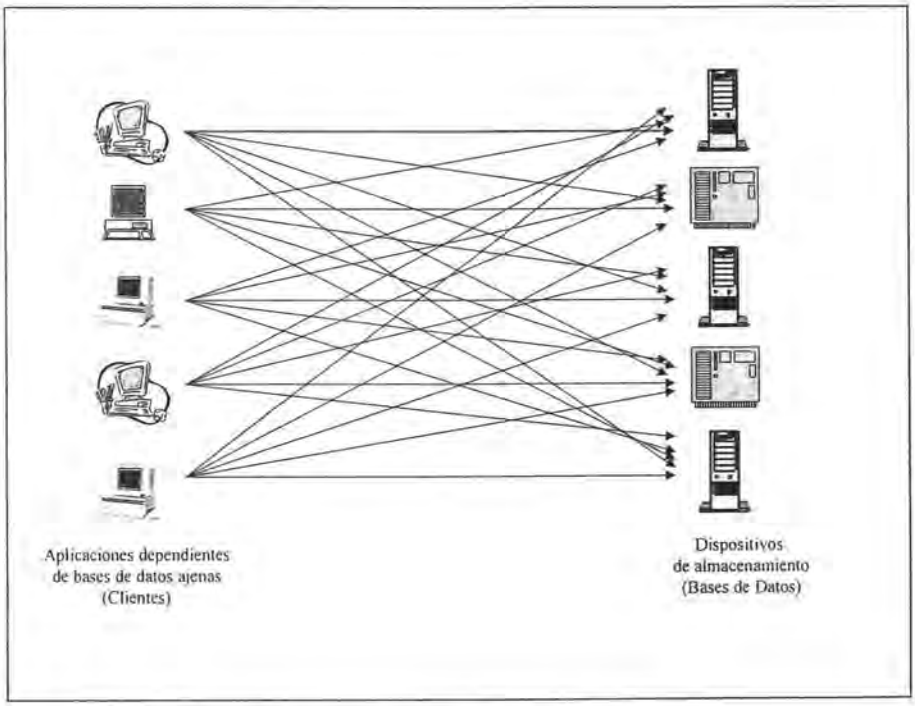

Figura 1.3 Modelo Cliente Servidor con interdependencias

Lo anterior motivó a la industria a generar una nueva opción para el desarrollo de aplicaciones empresariales, era bastante claro que la arquitectura ***Cliente Servidor*** funcionaba, pero también tenía sus limitantes, otra de estas limitantes era la poca reutilización de los componentes construidos lo que obligaba a los desarrolladores de software a programar en cada aplicación las mismas rutinas.

Como puede apreciarse, el poder de cómputo disponible no era muy grande, sin embargo estas aplicaciones basaban su éxito en que la lógica del negocio estaba en el cliente y aprovechaba el procesador de la PC para resolver las operaciones y actualizar al final en forma transaccional la base de datos.

Con este antecedente y tomando en cuenta el problema de la dispersión y crecimiento exponencial de los usuarios que con el advenimiento del protocolo ***HTTP*** se elevaban a cientos de miles, se cayó en la cuenta de que la arquitectura Cliente Servidor no podria soportar los nuevos requerimientos de la industria y era necesario contar ya con una nueva arquitectura, y es fácil ver porqué:

1.- La aplicación debía ser instalada en cada cliente.
2.- Poder de procesamiento del cliente no homogéneo.
3.- Conexiones limitadas en el servidor de base de datos.
4.- Licenciamiento limitado.
5.- Poco control sobre el software instalado en el cliente.
6.- Dificultad para replicar cambios o actualizaciones.

De estos problemas surgió la idea de agregar una nueva capa a la arquitectura Cliente Servidor, esta capa se conocería como la capa de Middleware o capa de Lógica de Negocio. De allí nace la nueva arquitectura que se conoce como Modelo Multicapas y que soporta al desarrollo de Aplicaciones Empresariales y que se representa en la siguiente figura.

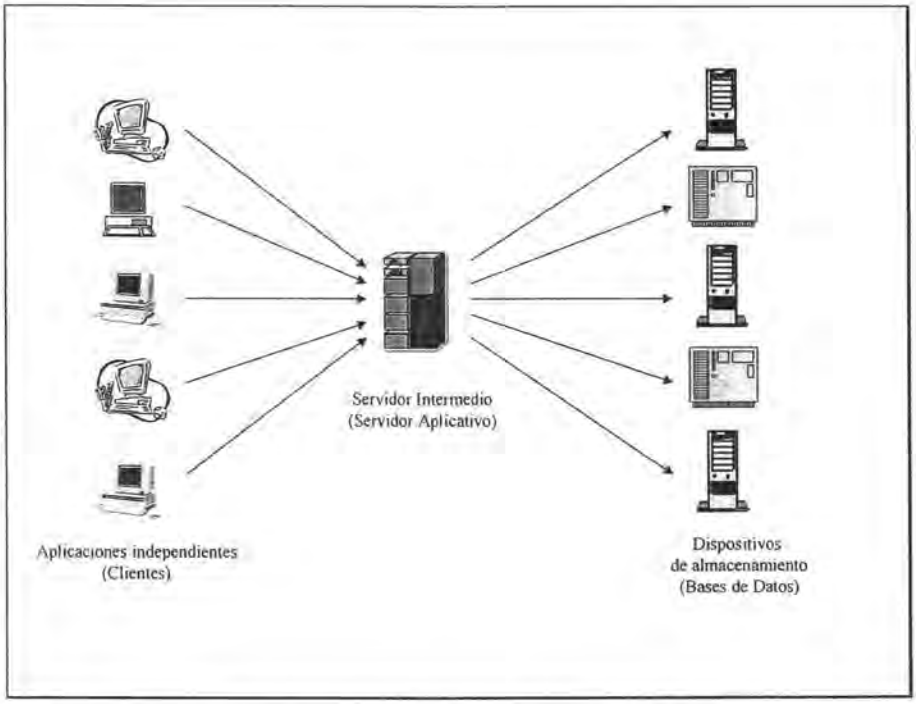

Figura 1.4 El Servidor Intermedio permite integrar aplicaciones independientes

Tecnología sobre la cual se implementan estas soluciones:

Cliente: Computadora Personal	**Servidor Intermedio:** HPUX Server E9000	**Servidor**: HPUX Server E1000
Procesador Intel Pentium I o superior.	Procesador RISC.	Procesador RISC.
166 Mhz o superior	600 Mhz	100 Mhz
64 MB en RAM	1GB en RAM	32 MB en RAM
Sistema Operativo: Windows 95 o superior	Sistema Operativo: Unix	Sistema Operativo: Unix
Software aplicativo: Visual Basic 6, Browsers o Clientes Java	Servidor Aplicativo y Monitores de Transacciones	Base de Datos: Sybase 12, Oracle.
	JDBC, Tuxedo, RPC, SSL, Java, JCA, etc.	Otras aplicaciones.

Este tipo de arquitectura representó un gran avance en el desarrollo de aplicaciones ya que permitía la reutilización de componentes, reducía el nivel de conexiones entre capas y otorgaba una mayor posibilidad de crear aplicaciones más robustas, escalables y con mayor funcionalidad.

Sin embargo también generaba mayores retos, era necesario crear algún mecanismo que se encargara de administrar los componentes, proveer conexiones hacia las diferentes bases de datos independientemente del proveedor y además resguardar de forma segura las aplicaciones. Por todo esto se pensó en crear un middleware que resolviera lo anterior y los demás requerimientos que una aplicación distribuida necesita, de lo anterior nacen los Servidores Aplicativos que apoyan a los Web Servers y que son el corazón de las aplicaciones empresariales distribuidas, estos ***Servidores Aplicativos (Aplication Servers)*** permiten no solo concentrar en un dispositivo intermedio mucha de la lógica de negocio, sino servir como integradores entre las diferentes aplicaciones, y esto lleva al nacimiento de nuevas aplicaciones que serán conocidas como las Aplicaciones Empresariales.

Pero el desarrollo de ***Aplicaciones Empresariales (Enterprise Applications)*** requiere el conocimiento de varias tecnologías, no es suficiente conocer un solo entorno de programación (como puede ser Java) para crear una aplicación de este tipo. Y aunque en este trabajo se tomará como base la tecnología Java, debe resaltarse el hecho de que el lenguaje Java no es solo una serie de instrucciones para codificar pasos a seguir con un propósito específico, por el contrario, Java y todo lo que gira entorno a él, debe entenderse como una tecnología completa y no como un lenguaje de programación más. De aquí la necesidad de comprender diferentes conceptos antes de entrar de lleno en este mundo de las Aplicaciones Empresariales, a las que además les agregaremos el término de Distribuidas.

El término distribuido puede aplicarse en muchos aspectos en el área de la computación, sin embargo, en este trabajo se usará como un auxiliar para describir el hecho de que una aplicación no se encuentra físicamente en una sola máquina, o que es atendida por un solo procesador, sino por el contrario que la aplicación se encuentra repartida y replicada entre diferentes capas. Este es el caso de las ***Aplicaciones Web (Web Applications)*** las cuales son instaladas en un servidor o arreglo de servidores para ser utilizadas por usuarios que pueden estar físicamente en el mismo edificio, en la misma cuadra, en la misma ciudad, en el mismo país, en el mismo continente e incluso en continentes diferentes.

Una tecnología que ha atacado el punto anterior es precisamente el protocolo ***HTTP (Hyper Text Transfer Protocol)***, el cual se basa en requerimientos, lo cual indica que no hay una conexión permanente entre quien provee el servicio (el servidor) y quien solicita el servicio (el cliente) puesto que se crea la conexión sólo por medio de una petición. Dado que no existe una conexión entre cliente y servidor se considera al HTTP como un protocolo sin estado (stateless) es decir que no guarda el estado de la conversación, ahora bien, cuando este protocolo comenzó a usarse, su propósito era transferir texto o lo que conocemos como páginas HTML (páginas estáticas), sin embargo, hoy ha evolucionado y es usado para diferentes propósitos como por ejemplo, el envío de archivos y la transferencia de información no estática.

HTTP fue una buena opción para las aplicaciones que se desplegaban desde un servidor remoto hasta el Browser ubicado en una PC de un usuario, sin embargo, el protocolo tiene una desventaja, esta desventaja consiste en la poca capacidad que se les da a los creadores de aplicaciones para realizar operaciones de negocio, es decir, las aplicaciones Web eran muy buenas para cuando se quería presentar información estática, como por ejemplo la presentación de anuncios o propaganda, pero es poco lo que se puede hacer

cuando se intentan crear aplicaciones que vayan mas allá y pretendan realizar algo mas que pura presentación.

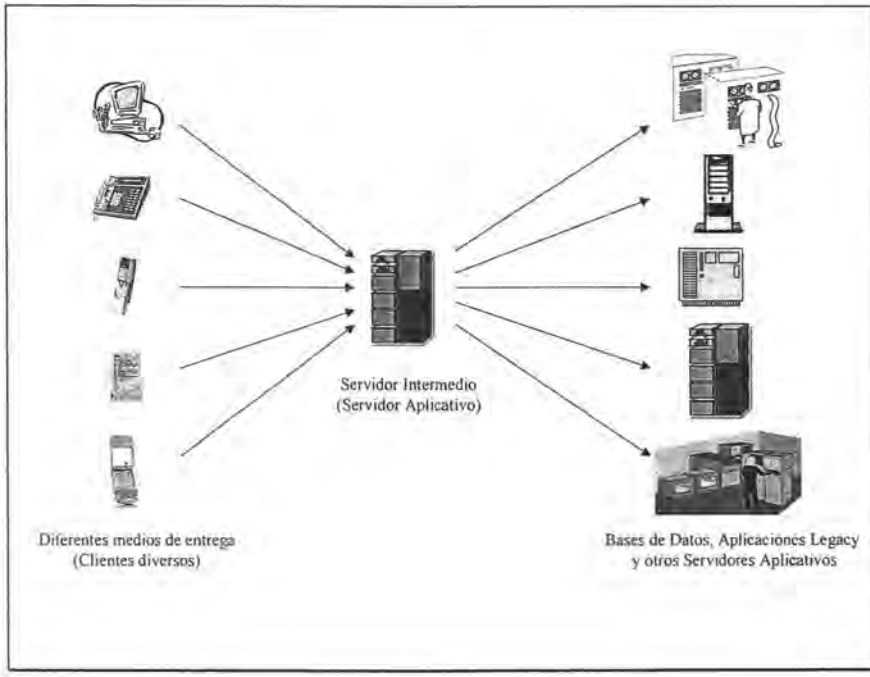

Figura 1.5 Componentes básicos usando un Servidor Aplicativo

Tecnología sobre la cual se implementan estas soluciones:

Cliente: Computadora Personal	**Servidor Intermedio:** HPUX Server E9000	**Servidor**: HPUX Server E1000
Procesador Intel Pentium I o superior.	Procesador RISC.	Procesador RISC.
166 Mhz o superior	600 Mhz	100 Mhz
64 MB en RAM	1GB en RAM	32 MB en RAM
Sistema Operativo: Windows 95 o superior	Sistema Operativo: Unix	Sistema Operativo: Unix
Software aplicativo: Visual Basic 6, Browsers o Clientes Java	Servidor Aplicativo y Monitores de Transacciones	Base de Datos: Sybase 12, Oracle.
	JDBC, Tuxedo, RPC, SSL, Java, JCA, etc.	Otras aplicaciones.
Cliente: Computadora Personal		
Minibrowser WAP		
Cliente: Palm		
Minibrowser WAP		

El diagrama anterior muestra el tipo de soluciones que pueden ser implementadas con ayuda de un Servidor aplicativo y la variedad de dispositivos que pueden conectarse; este trabajo toma como base una Aplicación Empresarial que servirá de apoyo para la descripción de la arquitectura y diseño de este tipo de aplicaciones, será esta aplicación la que se tome como ejemplo para detallar las capas, las piezas de cada una de ellas y las conexiones entre cada una de las capas, con este sistema ejemplo un usuario podrá consultar sus saldos desde diferentes medios de entrega, por ejemplo, un Teléfono Celular, desde una Palm o desde un Web Browser. Esto es lo que se pretende realizar con este trabajo, explicar de manera clara y concisa como se pueden construir Aplicaciones Empresariales basadas en ***J2EE (Java 2 Enterprise Edition)*** utilizando básicamente las capas de Presentación, Lógica de Negocio y Datos.

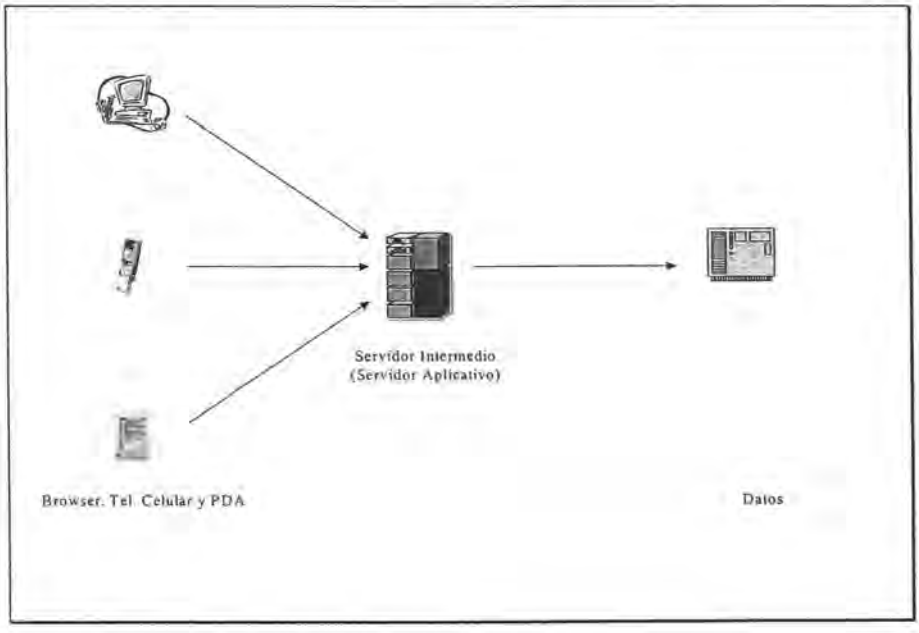

Figura 1.6 Esquema de una aplicación tipo

Ahora que se ha descrito lo que es una aplicación de negocio, hablaremos de las aplicaciones distribuidas. El objetivo principal de una ***Aplicación Distribuida*** es manejar de mejor manera la complejidad y el costo de las aplicaciones, haciéndolas altamente disponibles, escalables y fáciles de mantener. Las aplicaciones distribuidas pueden hacer esto al utilizar muchos y más simples sistemas auto contenidos que trabajan en conjunto para realizar la misma funcionalidad que haría un solo sistema.

1.2 Aplicaciones Distribuidas

En general, las Aplicaciones Distribuidas se componen de pequeños componentes que son ágiles, y tienen una gran versatilidad para adaptarse a los cambios, esto al contrario de las aplicaciones monolíticas (cliente / servidor) los cuales son generalmente grandes, lentos y no se adaptan a los cambios.

Las Aplicaciones Distribuidas tienen las siguientes características:

A) Las aplicaciones distribuidas dividen el trabajo en diferentes módulos independientes.

B) El error en uno de estos módulos tiene menor impacto en la totalidad del sistema, por ello las aplicaciones distribuidas son:

- *Disponibles*
- *Escalables*
- *Mantenibles*

La **disponibilidad** de un sistema es la cantidad de tiempo que puede estar procesando requerimientos de un cliente.

Se dice que un sistema tiene **alta disponibilidad** cuando puede estar atendiendo peticiones de usuario (up & running) prácticamente las 24 horas del día, los 365 días del año. La alta disponibilidad se logra usando técnicas de balanceo y error/relevo (Tolerancia a fallas).

En este trabajo, nos referiremos con el termino de Aplicaciones de Empresariales o Aplicaciones de Negocio a aquellas aplicaciones que necesiten Alta Disponibilidad, ejemplos de estas aplicaciones son: Cajeros Automáticos, Servicios Bancarios en Internet, Servicios de Inventario en tiempo real, sistemas de monitoreo y en general cualquier aplicación que por sus características necesite estar disponible todo el tiempo.

La **escalabilidad** de un sistema es la habilidad de crecer y poder manejar la demanda de trabajo (peticiones de usuario). La escalabilidad es importante debido a que una máquina solo puede realizar una cantidad finita de trabajo; en el mercado es común que en si el negocio tiene éxito, la demanda crezca y si esta capacidad sobrepasa los límites de trabajo de la máquina en donde se encuentre nuestro sistema nos vemos forzados a distribuir el trabajo en diferentes máquinas y si el sistema no consideró esto en su diseño resulta muy difícil lograr la escalabilidad.

Se dice que un sistema es **mantenible** cuando su complejidad se maneja de mejor manera y resulta en qué tan fácil ese sistema puede ser actualizado, debugeado o mantenido. Esta característica tiene un impacto directo en qué tan rápidamente un sistema puede cambiar o adaptarse a los requerimientos. Esto se logra colocando la lógica de negocio en componentes modulares y reusables. Además también se logra poniendo la lógica de infraestructura en servicios independientes de la lógica de negocio, estos servicios son el acceso a base de datos, seguridad, monitoreo de transacciones, etc.

Un **cliente aplicativo** es cualquier cliente que pueda comunicarse con un Sistema Distribuido. Estos clientes pueden ser Clientes Pesados o Clientes Ligeros, esta clasificación se hace de acuerdo a la cantidad de procesamiento de negocio que se realiza localmente en el cliente.

En este trabajo estaremos apoyándonos en el estándar J2EE para la creación de este tipo de aplicaciones, y lo haremos porque cualquier desarrollo puede tomar muchas ventajas de los estándares ya que estos hacen una separación de los problemas y plataformas, además permiten la modularización de la construcción de software y nos dan un enfoque orientado a la resolución de problemas que ya se han resuelto en la industria.

1.3 La Integración de Aplicaciones

Uno de los principales retos a los que se enfrenta un arquitecto de sistemas es el de tener que lidiar con diferentes plataformas, diferentes sistemas operativos, diferentes lenguajes de programación y hasta con diferentes empresas, ***ese es el punto medular del porqué de este trabajo,*** hoy en día existen múltiples tecnologías con las que se pueden resolver los diferentes requerimientos que se les presentan a las empresas y a sus áreas de negocio. Dada esta gran variedad un arquitecto o diseñador de sistemas se enfrenta al dilema de *"¿qué tecnología debe usarse?, ¿cuál es la tecnología que resolverá más eficientemente la problemática?"*, estas preguntas no son fáciles de responder, el arquitecto debe prever el posicionamiento de dicha tecnología, el soporte, si existe un distribuidor o gente especializada en el país, el costo, el tiempo de vida de la tecnología, su madurez, el desempeño y hasta la facilidad de uso; todo esto sin contar con el análisis que debe hacer de las aplicaciones con las cuales tenga que interactuar y el futuro del sistema mismo.

Como puede verse, no se trata de tomar decisiones fáciles, son decisiones que no pueden tomarse a la ligera ya que de ser así se compromete el prestigio de la compañía, recordemos que hablamos de aplicaciones empresariales en donde una caída del sistema o retraso del mismo puede generarle a los usuarios inconvenientes muy severos que repercutirían directamente en el prestigio y futuro de la compañía.

Por lo anterior, la ***Integración de Aplicaciones (Enterprise Aplication Integration, EAI)*** toma una gran importancia ya que se debe determinar la forma correcta en que las aplicaciones deben interactuar, y esto debe pensarse en función del mejor desempeño, mejor escalabilidad, recursos y tecnologías disponibles, presupuesto, futuro del servicio y por supuesto del margen de tiempo para desarrollar la solución.

El arquitecto de sistemas debe saber decidir correctamente entre seleccionar soluciones inmediatas contra soluciones a largo plazo o con una visión futurista. Bajo este concepto lo más común es tomar la decisión que solucione rápidamente el problema, es decir, optar por la solución inmediata, pero si todas las soluciones que se tomen son para resolver problemas específicos sin ver el contexto general entonces caemos en lo expuesto en la sección de antecedentes en donde se planteó la problemática de tener aplicaciones monolíticas y no aplicaciones distribuidas.

Entonces, un arquitecto de sistemas no solamente debe tomar decisiones de qué funcionalidad debe poner en cada capa, ni la tecnología a usar, sino también la forma en que los diferentes componentes deben interactuar para lograr una correcta integración entre aplicaciones y no estar "alambrando" sistema a sistema cada vez que se necesita. Lo anterior podria parecer no tener mucha importancia pero en compañías en donde la información y los procesos automatizados tienen el control del ciclo de vida de la producción, una mala decisión puede llevar a pérdidas millonarias, y no solo por equivocarse en la tecnología a usar, si no por la poca o nula reusabilidad de piezas y componentes que deben ser construidos específicamente para cada requerimiento, elevando los costos de construcción y por ende de operación, además de que a la larga se tendrá que revisar la solución para desecharla y hacer algo con una mejor integración.

Es por esto que en este trabajo también se presenta y recomienda el uso del Servidor Aplicativo y Monitor de Transacciones, que si bien no es el objetivo de este trabajo si permitirán apoyarnos en la definición de la metodología que se propondrá.

Ahora bien, aunque la integración de aplicaciones no es en sí una tecnología, si es una tendencia tecnológica que pretende interconectar las diferentes aplicaciones que pueden coexistir en un negocio o hasta varios negocios independientes entre si, este último comentario es el que nos lleva a decidir la tecnología base sobre la cual trabajaremos:

- Sistema Operativo Unix 11.0.
- Sistema Microsfot Windows 98 o superior.
- Java 1.31.
- Web Server (iPlanet 3.6 o superior).
- Clientes pesados (Visual Basic 6.1 o superior).
- Microsoft Web Browser 5.0, Browser Netsacape 3 o superior.
- WebLogic Server 6.1 Service Pack 3 o superior.
- Tuxedo 6.5 o superior
- Protocolo http para producción y protocolo t3 para configuración.
- WML

La lista anterior es solamente una lista de las tecnologías y plataformas sobre las que mayor experiencia se tiene, sin embargo la integración de las aplicaciones, componentes, plataformas, protocolos y clientes no esta cerrada a la lista anterior, únicamente se da un marco introductorio de lo que se manejará en este trabajo.

1.4 La visión Java 2 Enterprise Edition

En este trabajo se utilizará la visión del J2EE debido a que este estándar ayuda a crear aplicaciones que son:

- Estandarizados
- Escritos en Java
- Pueden correr en cualquier servidor aplicativo o incluso Web Server
- Son abiertos y fáciles de integrar a otras tecnologías.

Sun Microsystems ha definido esta especificación para clasificar plataformas y servidores aplicativos e incluye estándares para servicios como HTTP, HTTPS, Java Transaction API, RMI-IIOP, Java IDL, JDBC, Java Message Service, Java Naming and Directory interfase, Java Mail, y Java Beans, estas tecnologías serán presentadas más adelante, pero por ahora basta decir que son el conjunto de Aplication Program interfase (API's) con los que las Aplicaciones Empresariales y distribuidas de las que hemos hablado serán construidas.

Aún cuando no es intención de este trabajo presentar todas las tecnologías que proporciona J2EE, y tomando en cuenta que dicho estándar se encuentra en evolución al momento de escribir este trabajo, se mencionará y dará una breve explicación de las API's más usadas de J2EE.

(API)	Descripción
Java Servlets	Esta tecnologia provee el mecanismo principal para implementar lógica de presentación ya que los Java Servlets son programas que permiten codificar en lenguaje java la respuesta al usuario y al mismo tiempo dan la flexibilidad de tener toda la lógica que un lenguaje de programación proporciona. Por ejemplo, un Java Servlet tiene mayormente lógica (codificación Java) pero permite combinarla con código de presentación (HTML por ejemplo).
Java Server Pages (JSP)	Al contrario que los Java Servlets, un JSP fue creado para proporcionar un mecanismo en donde la codificación de la presentación (HTML por ejemplo) fuera más fácil, dado su diseño los JSP facilitan la presentación permitiendo combinarla con lógica, por ejemplo: facilitan la codificación de la presentación (HTML) y permiten combinarlo con lógica (codificación Java).
Java Beans	Los Java Beans son clases java que proporcionan un mecanismo para codificar funcionalidad típica de librerias, sin embargo también son utilizados para almacenar datos en memoria. Generalmente son utilizados como auxiliares para cumplir tareas específicas (funciones) y para guardar temporalmente datos.
Java Data Base Conectivity (JDBC)	JDBC es un mecanismo que proporcionan los proveedores de bases de datos para que las aplicaciones J2EE puedan conectarse a sus motores de bases de datos, JDBC tiene funcionalidad para consultar la base de datos, para modificarla y para hacer transacciones sencillas y complejas (Two Phase Commit).
Java Transaction API (JTA)	JTA permite que las aplicaciones J2EE puedan coordinar transacciones ya sea en una sola base de datos o en un conjunto de varias bases de datos, efectuando las operaciones de escritura cuidando que todas las bases permanezcan integras, además de que permite que las operaciones sobre la base de datos sean repetibles y consistentes. Aprovecha también el Two Phase Commit (2PC) para efectuar y coordinar transacciones entre dos o más bases de datos.
Java Naming and Directory interfase	JNDI es un repositorio central en donde se encuentran referencias a todos los componentes que se encuentran publicados dentro de un Servidor de Aplicaciones, de esta manera una aplicación puede reutilizar

(JNDI)	componentes que ya se encuentren en el JNDI. Generalmente este JNDI es conocido como Árbol JNDI ya que en el se registran las referencias a todos los recursos disponibles, así mismo controla el que no se dupliquen sus nombres.
Remote Method Invocation (RMI)	RMI es la tecnología que permite a la parte cliente efectuar llamados a objetos java remotos, estos objetos exponen su interfase en manera de métodos que son exportados para ser llamados. Estos objetos aparentan estar localmente al cliente pero en realidad se encuentran en un servidor aplicativo, esto es posible lograrlo gracias al JNDI.
Enterprise Java Beans (EJB)	Los EJB son el estándar que proporciona J2EE para implementar la lógica de negocio, es decir, son objetos cuya función principal es la de resolver el negocio, así como los JSP y Servlets se encargan de la presentación, los EJB se encargan de la lógica que proporciona la información a ser presentada en la capa de presentación. La tecnología java ofrece a los desarrolladores la posibilidad de pensar tan solo en implementar la lógica de la aplicación y dejar al servidor de aplicaciones la tarea de resolver los mecanismos de infraestructura como seguridad y administración de los recursos.
Stateless Enterprise Java Beans	Un EJB puede ser usado como un resolutor de operaciones puntuales que no necesiten conocer el contexto del cliente, es decir, operaciones de negocio que independientemente del cliente que las esté usando regresan el mismo tipo de resultado.
Stateful Enterprise Java Beans	Los EJB's también pueden implementarse de manera conversacional, es decir, el EJB puede atender a solo un cliente a un tiempo, tener una sesión hacia su cliente y mantener el estado de la conversación con dicho cliente, esto permite que operaciones complejas de negocio sean resueltas por un componente que esta ligado a un cliente mientras este mantenga una sesión activa.
Entity Enterprise Java Beans	Existe otro tipo de EJB que fue creado para interactuar con una base de datos, la finalidad de este tipo de EJB es la de mantener actualizada y disponible la información que se encuentra en una relación de tablas, ocultando lo más posible la conectividad hacia la base de datos. Estos EJB cuentan con mecanismos especiales para efectuar este tipo de tarea.
Message Driven Enterprise Java Beans	Al evolucionar el estándar J2EE se descubrió la necesidad de contar con mecanismos asincronos, ya que las todos los mecanismos inicialmente propuestos eran síncronos y existían necesidades que las aplicaciones basadas en J2EE no podían resolver dada la falta de tecnología "off-line". De esta manera nacen los Message Driven EJB's que activan sus mecanismos en base a mensajes provenientes de colas de mensajes tipos JMS, así, cuando un mensaje llega a la cola destino un método es activado en el EJB.
Java Message Service (JMS)	JMS es el estándar que proporciona J2EE para resolver los mecanismos de mensajería entre componentes. La especificación JMS provee a los desarrolladores el API para implementar servicios como colas de mensajes, publish & subscribe además de otras tecnologías como push & pull. De esta manera un objeto (cliente) puede emitir mensajes o solicitudes de servicio a otros componentes que efectuarán el trabajo sin la necesidad de que el cliente espere a la finalización de la tarea ya que posteriormente puede ser notificado del resultado de la operación.

Java Mail API	Java Mail es la interfase que provee la posibilidad de conectarse a muchos diferentes tipos de aplicaciones de correo electrónico, usa un protocolo orientado a objetos y contiene las clases que soportan el correo electrónico.
Java Authentication and Autorization Service (JAAS)	El JAAS es un modelo propuesto por J2EE para autentificar a los clientes y darles niveles de acceso a los recursos publicados en el servidor aplicativo.

Lo anterior solo es una muy breve descripción de los principales API's Java existentes, y aunque día con día estas implementaciones del estándar son mejoradas la idea básica se mantiene: contar con una serie de reglas y especificaciones para que una aplicación creada con estas características sea completamente portable de una plataforma a otra, siendo esta una característica muy importante en el desarrollo de aplicaciones empresariales.

La siguiente ilustración muestra la división de múltiples capas que este estándar pretende seguir, la idea como se puede ver continua siendo la misma: separar la funcionalidad de la aplicación en diferentes capas, que no necesariamente son físicas, sino lógicas, y el estándar J2EE ayuda a lograr este propósito.

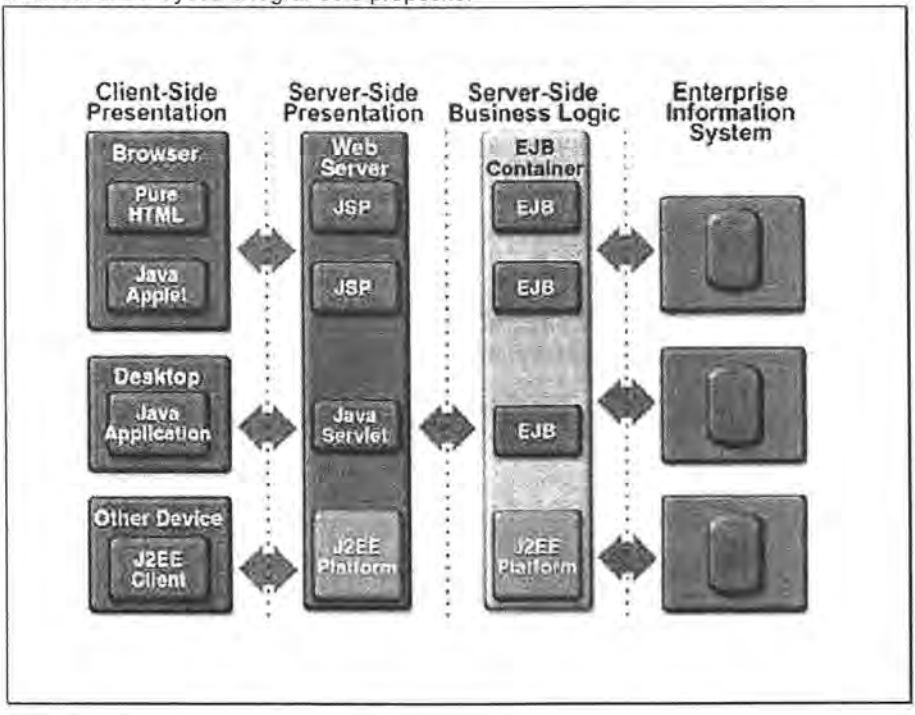

Figura 1.7 J2EE y el modelo de múltiples capas

El estándar J2EE evoluciona constantemente y constantemente se le están agregando más definiciones y API's, esto lo hace un estándar internacional que responde eficientemente a las necesidades del mercado, sin embargo en este trabajo solo se mencionan algunos de los API's para comprensión general, si el lector desea mayor información deberá recurrir a una fuente especializada en dicho API.

La información anteriormente descrita se basa en los modelos o patrones de procesamiento, los cuales se utilizan para acelerar el desarrollo de una aplicación, sin embargo, un patrón de diseño es una idea abstracta para aprovechar un conjunto de clases que tienen un objetivo en común. Estos patrones ayudan a resolver problemas de diseño particulares, aunque un patrón de diseño no es directamente reutilizable a menos que se usen herramientas que ayuden a la construcción de sistemas basados en patrones. Los patrones ayudan a integrar en las aplicaciones conjuntos de elementos que trabajan en común para lograr una tarea ya que la idea principal es la de tratar de construir sistemas basados en la agrupación de componentes en lugar de escribir líneas de código dentro de un solo programa.

Los patrones dan a los componentes las siguientes características importantes dentro de la construcción de sistemas distribuidos:

- Deben estar encapsulados e intercambiar información solo mediante el uso de una interfase bien definida.
- Deben usarse en conjunto.
- Deben estar organizadas en capas y cada capa representando una familia de componentes.

Aún con estas ventajas, el utilizar patrones presenta el reto de organizar los componentes y con ello construir la aplicación, por lo que en el siguiente capítulo se describe una propuesta de organización en capas para resolver este problema.

CAPÍTULO 2
EL MODELO DE MÚLTIPLES CAPAS

2.1 La división en capas

La principal característica de este modelo consiste en la separación de la funcionalidad que compone a una Aplicación Empresarial Distribuida en capas. Esta funcionalidad está dividida en las capas que se presentan en la figura 2.1, en donde claramente se observa la separación de responsabilidades.

Figura 2.1 Modelo de N Capas

En la tabla 2.1 se muestra la correspondencia de cada capa y su funcionalidad.

CAPA	FUNCIONALIDAD
Presentación de Usuario	La capa de presentación es la interfase con el usuario, es el punto de contacto con este, pudiendo ser inclusive el contacto con otra aplicación, y es la entrada y salida de información.
Presentación de Negocio	Es la capa que se encarga del apoyo a la presentación, proporciona datos para la presentación y resuelve tareas que no implican transaccionalidad.
Integración de funcionalidad de Negocio	Esta capa es la que implementa todas las reglas del negocio, es decir, resuelve la transaccionalidad que involucra la actualización de los datos.
Acceso a datos	Esta capa contiene la funcionalidad para consultar, actualizar y borrar los datos de manera unitaria, se caracteriza por no tener control transaccional.
Datos	Son los datos empresariales en si y las reglas de replicación de estos. Este mecanismo garantiza la separación de las estructuras y manejadores de base de datos de la aplicación de reglas.

Tabla 2.1 Modelo de N Capas

En el modelo de la figura 2.1 se observa que cada capa puede invocar a la capa inmediata siguiente y el flujo del requerimiento corre por todas las capas y retorna de la misma forma. La capa de Integración de Funcionalidad de Negocio puede invocar servicios de ella misma además de servicios de Acceso a Datos. Una regla importante es que el salto de capas no es permitido, cada capa solo puede estar en contacto directo con su capa predecesora o con su capa antecesora, por ejemplo, la capa de presentación no puede interactuar directamente con la capa de acceso a datos, debe pasar primero por la capa de Integración de Funcionalidad de Negocio, es decir siempre debe invocarse a la capa inmediata, esto se debe a que de esta manera, el modelo protege la integridad de los datos y las reglas de negocio, con la finalidad de poder contar con elementos reusables que permitan un ahorro en los costos de desarrollo de aplicaciones posteriores. Continuando con el análisis al modelo, se detalla en la figura 2.2 que en nuestro ámbito la capa de Integración de Funcionalidad de Negocio puede dividirse en tres capas adicionales.

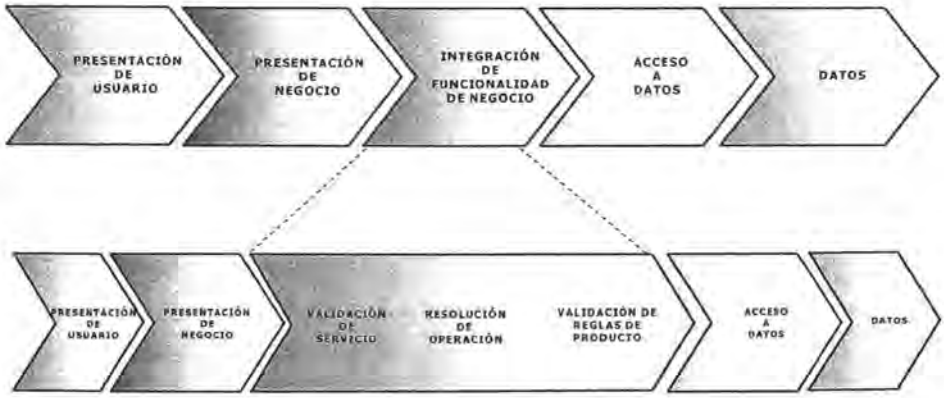

Figura 2.2 Integración de Funcionalidad de Negocio

De esta manera se obtiene la tabla 2.2 en donde se especifica la funcionalidad de cada subcapa:

SUBCAPA	FUNCIONALIDAD
Validación del servicio	Asegura que cualquier operación solicitada tiene garantía de haber pasado por procesos de validación corporativos y esta preparada para ser procesada. Por esta razón esta es la única capa a la que tiene acceso las capas de Presentación de Negocio
Resolución de la operación	Permite la aplicación de reglas de negocio que resulta en la descomposición de operaciones en transacciones. Esta capa coordina las acciones, aplica reglas de negocio y asegura la integridad transaccional.
Validación de Reglas de producto	Aplican las políticas del producto previas a la actualización de los datos. Es decir, aplica reglas especificas del producto solicitado.

Figura 2.2 Integración de Funcionalidad de Negocio

Otro punto importante es que estas capas pueden hacer uso de la capa de Acceso a datos para complementar información antes o después de la resolución o para solicitar ejecución de transacciones.

Lo anterior es la base para generar los ***Patrones de Procesamiento*** que son un conjunto de técnicas y reglas de organización lógica y física que permiten moverse de un concepto a un diseño y a una implantación.

El propósito de identificar y documentar estos patrones es construir las instrucciones y cálculos necesarios para traducir un requerimiento de negocio o de productividad a patrones preestablecidos que permiten la reusabilidad y homogeneización de los diversos componentes. Así mismo, facilitan la planeación de soluciones y la obtención de métricas de costo y duración de proyectos.

Los patrones no son rígidos, y su uso debe ser flexible, esto quiere decir que por cada patrón pueden existir varios escenarios que describan ligeras variantes físicas del Patrón General. Considerando lo anterior, de manera general deberá utilizarse el escenario que ofrezca las últimas tecnologías disponibles para generar la arquitectura final deseada.

Sin embargo existen otros factores que influyen en la selección del escenario a utilizar, por ejemplo, el medio de acceso por el cual se ofrecerán al usuario los servicios de la aplicación, otro factor es que el escenario Internet es diferente al de Intranet ya que están involucrados aspectos de seguridad. Debe recordarse también que uno de los objetivos de los Patrones de Procesamiento es la reutilización de código y las vistas tecnológicas de los Patrones de Procesamiento incluyen los ***Componentes Arquitectónicos*** que son los elementos básicos sobre los cuales se implementa la aplicación, estos deben ser identificados ya que serán los que se reutilizarán en el proyecto y por medio de los cuales se lograrán los ahorros en desarrollo y se reforzará la integridad de la aplicación.

Es aquí en donde comienza la primera parte de la aportación y la propuesta de este trabajo, ya que a partir de este momento se presentarán los componentes que se consideran necesarios para crear una aplicación empresarial, contribuyendo además con estos componentes a practicar la reusabilidad.

Con lo anterior en mente no se debe olvidar que para cada uno de estos componentes es importante identificar si se cubre o no la funcionalidad esperada en las versiones liberadas o si será necesario hacerles modificaciones ya que este punto impacta de manera importante en los tiempos del proyecto.

Algunos de los componentes arquitectónicos identificados no sufrirán modificaciones, pero si será necesario cambiarles algunos de sus parámetros de configuración y deberán existir procedimientos y reglas para modificarlos ya que al ser reusables y utilizados por más de un sistema se pueden tener problemas serios si la modificación no es analizada, planeada y autorizada por cada uno de los consumidores del componente. Así mismo en la arquitectura del proyecto también habrá componentes arquitectónicos no mostrados en los patrones de procesamiento y deberán ser construidos dependiendo de la funcionalidad del proyecto, pero su construcción y desarrollo deberá hacerse siempre con un enfoque de reutilización y todos los elementos a ser construidos se deberán de ubicar en la capa funcional que les corresponda.

El modelo de N capas es una vista lógica general de la funcionalidad de las aplicaciones empresariales, sin embargo debe adecuarse a las necesidades especificas de las organizaciones, por ello es aquí donde comienza la propuesta de este trabajo, la cual comenzará a desarrollarse a partir del capítulo 3.

2.2 Patrón General: Las capas funcionales

Antes de poder armar la arquitectura de una aplicación distribuida basada en múltiples capas es necesario conocer los conceptos y componentes arquitectónicos básicos así como las relaciones permitidas entre ellos.

El objetivo de este patrón general es dar a conocer las responsabilidades de cada capa funcional, es decir, se describen cuales son las funciones de cada capa para que el usuario pueda diferenciar la responsabilidad de estas y ubicar los componentes de estas. Este patrón general se presenta en la figura 2.3.

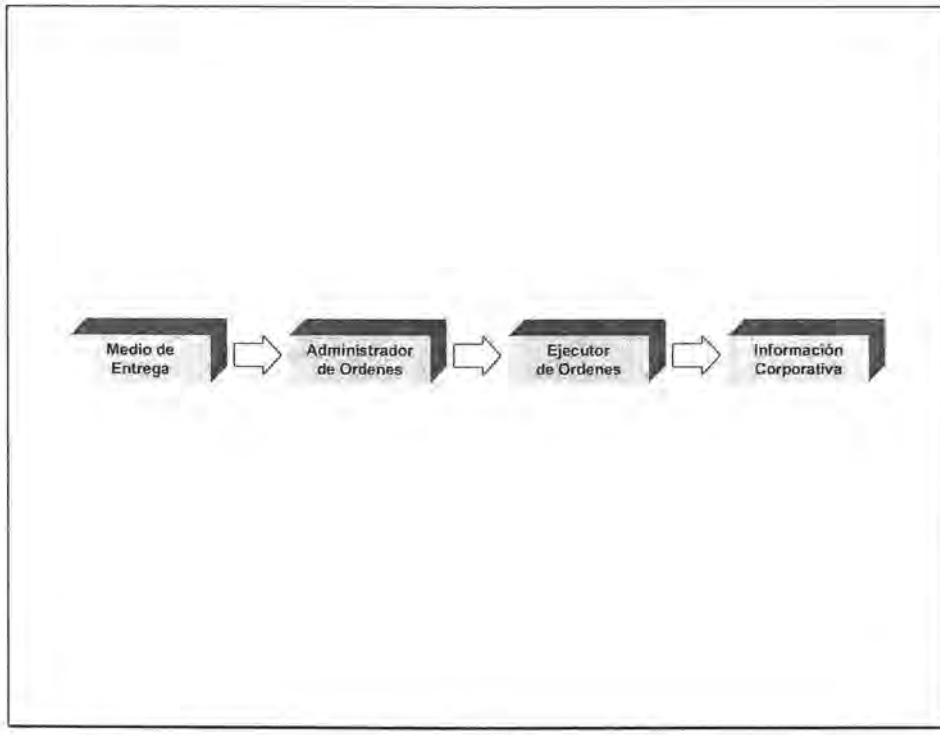

Figura 2.3 Patrón General

2.2.1 La capa del Medio de Entrega

La capa del Medio de Entrega es el mecanismo que tienen los usuarios para acceder a los servicios que proporciona el negocio. La ubicación de esta capa se muestra en la figura 2.4.

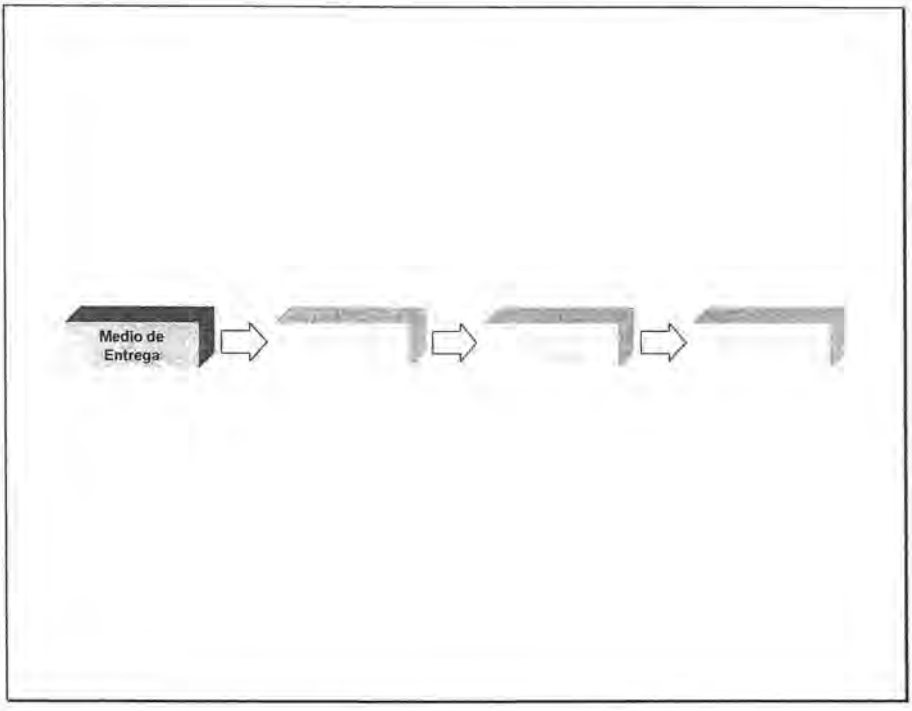

Figura 2.4 La capa del Medio de Entrega

Las responsabilidades de esta capa son:

- Presentar los servicios del negocio en términos del usuario.
- Proporcionar ayudas al usuario para que introduzca la información necesaria para enviar una orden.
- Validar sintácticamente la información proporcionada por el usuario.
- Monitorear la actividad del usuario.
- Manejar diversos dispositivos tecnológicos para facilitar y apoyar el acceso al usuario, es decir, tiene la posibilidad de interactuar con el usuario mediante varios tipos de dispositivos.
- Permitir al usuario llevar un control y registro de sus peticiones.
- Hacer amigable la interacción con el usuario controlando la presentación y orden que ésta se hace al usuario.
- Encriptar los datos sensitivos del usuario y de la solicitud.

- Permitir entregar avisos, anuncios e información mercadotécnica al usuario.
- Proporcionar tutoriales y ayudas para el usuario.
- Realizar la interpretación de la orden hecha por el usuario en sus términos para que pueda ser ejecutada en los términos del Negocio.

Esta capa esta condicionada a que no resuelve las órdenes de servicios del usuario, además de ser la interfaz para recibir órdenes desde otros negocios y solamente puede interactuar con la capa de administración de órdenes para hacerle solicitudes, por otra parte esta capa tiene el control mientras el usuario esta armando la orden y lo cede a la siguiente capa cuando hace la solicitud. En resumen su funcionalidad se centra en dar la presentación, navegación, validación sintáctica, validaciones entre datos y el acceso e interacción con el usuario.

2.2.2 La capa de Administración de Ordenes

La capa de Administración de Ordenes es la encargada de recibir, identificar, validar y registrar las órdenes del usuario. Además de administrar y controlar el flujo de ejecución de dichas órdenes. La ubicación de esta capa se muestra en la figura 2.5.

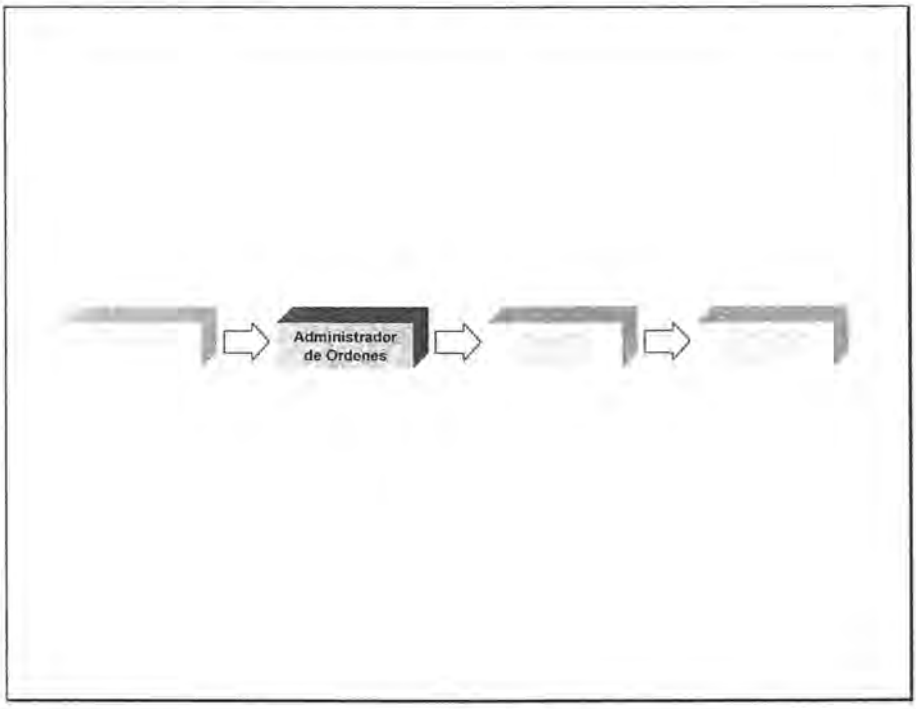

Figura 2.5 La capa de la Administración de Ordenes

Las responsabilidades de esta capa son:

- Identificar la orden y el medio de entrega que está usando el usuario.
- Autentificar al usuario y su perfil.
- Complementar los datos que por razones de ergonomía no haya incluido el usuario al mandar la orden.
- Validar que la orden se esté pidiendo en los términos en que lo establece la relación entre el usuario y el Negocio (contratos, clausuras, normatividad).
- Registrar las órdenes de cada usuario y medio de entrega.
- Aplicar mecanismos para la no-repudiación de las órdenes.
- Controlar la orden durante el ciclo de su atención y administrar la invocación a los diversos ejecutores de órdenes.
- Identificar las operaciones a ejecutarse en una orden y el flujo en que estas deben ejecutarse.
- Aplicar procesos previos y posteriores a la orden y que son necesarios para la operación del Negocio.
- Proporcionar al medio de entrega la información resultante de la ejecución de la orden.
- Permitir que una orden considere la integración de varios servicios básicos del negocio.
- Contar con mecanismos para la ejecución de las órdenes dependiendo de los eventos de interés que haya indicado el usuario.

Esta capa esta condicionada a que es la interfaz para solicitar órdenes a otros negocios, es la única capa que le hace solicitudes a la capa de ejecución de órdenes y además tiene el control durante todo el ciclo de vida de la orden. En resumen esta capa es la encargada del facultamiento, registro, autenticación, flujo de la orden, reglas de uso del servicio, conversión de datos, interpretación de errores y de implementar las reglas de intercambio con entidades externas.

2.2.3 La capa del Ejecutor de Ordenes

La capa del Ejecutor de Ordenes es la encarga de identificar, controlar y ejecutar las acciones implícitas en la resolución de las órdenes solicitadas por los usuarios. La ubicación de esta capa se muestra en la figura 2.6.

Las responsabilidades de esta capa son:

- Identificar las acciones a ejecutarse en una operación.
- Solicitar, ejecutar y controlar el flujo de ejecución de las acciones monitoreando el resultado de cada una de ellas.
- Aplicar los flujos de excepción o alternos a la ejecución de la operación dependiendo del resultado de las acciones que la componen.
- Tiene la facultad de solicitar la desaplicación de una, algunas o todas las acciones que componen una operación dependiendo del resultado de la ejecución o a petición del usuario.

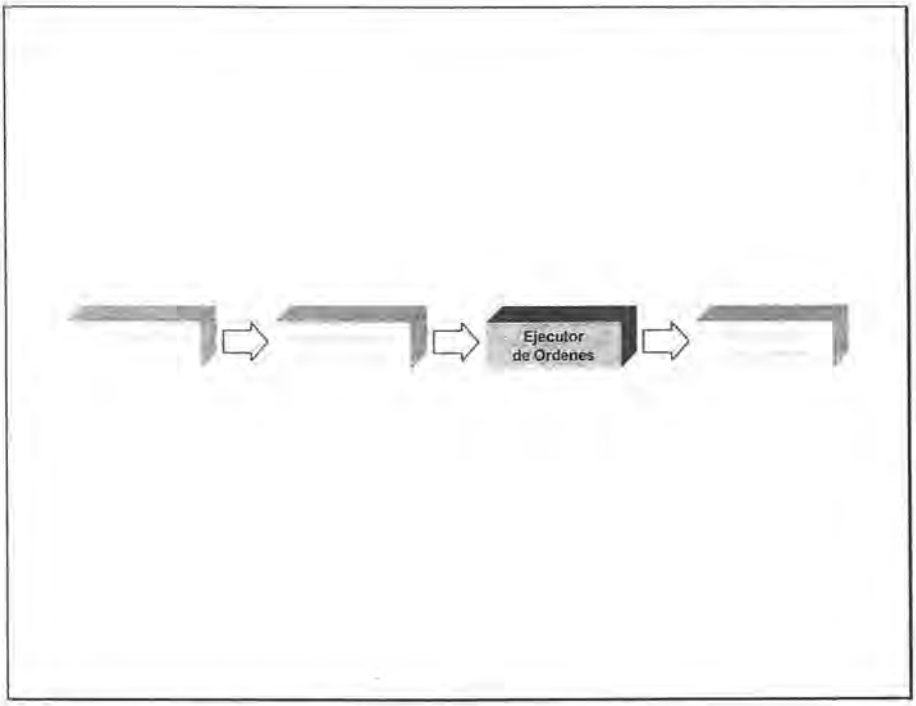

Figura 2.6 La capa del Ejecutor de Ordenes

• Aplicar los criterios necesarios para ejecutar en forma masiva, en grupo o individual las operaciones incluidas en una orden.

Esta capa esta condicionada a que no modifica directamente la información de los usuarios, del Negocio o de las propias solicitudes, es además la única capa que interactúa con la capa de información corporativa y tiene el control mientras se está ejecutando el flujo de la operación. En resumen esta capa es la encargada de implementar la integridad transaccional, los reversos, el flujo de la operación, el registro de la operación y la identificación de errores.

2.2.4 La capa de la Información Corporativa

La capa de Información Corporativa es la encargada de administrar y actualizar los datos institucionales. Siendo estos datos de toda índole, es decir son los datos de los usuarios, del negocio, de la relación entre los usuarios y el negocio, de la operación del usuario y de la operación del negocio, la información de gestión y administración interna así como la información y estadísticas derivadas de éstos. La ubicación de esta capa se muestra en la figura 2.7.

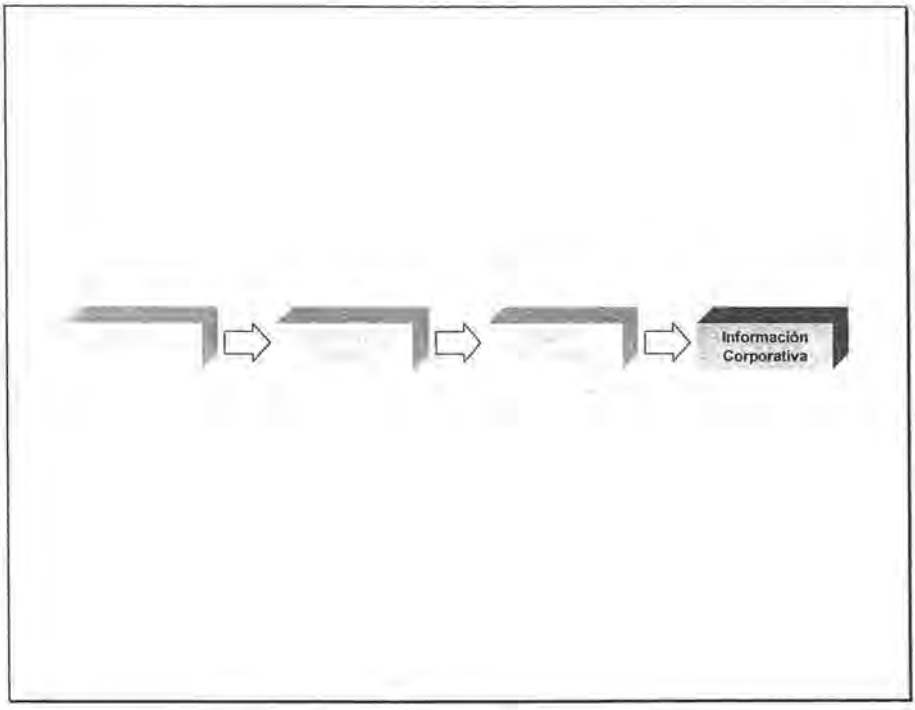

Figura 2.7 La capa del Información Corporativa

Las responsabilidades de esta capa son:

- Ejecutar las acciones contenidas en las operaciones, estas acciones se derivan en actualizaciones a los datos institucionales.
- Registrar las actualizaciones a los datos institucionales.
- Aplicar las reglas de integridad derivadas del modelo institucional de datos del negocio.
- Desencriptar los datos sensitivos del usuario y de la solicitud.

Esta capa esta condicionada a tener el control mientras se está haciendo la actualización a los datos institucionales. En resumen esta capa es la encargada de implementar las reglas aplicables a los productos y servicios, registro contable, integridad del modelo de datos y validación semántica de las actualizaciones de datos.

Ahora que se conoce las responsabilidades de cada capa de forma general, es posible pasar a ver el detalle de los componentes que se ubican en cada capa.

2.3 El Modelo de Múltiples Capas

2.3.1 Los componentes lógicos de las capas y su implementación física

Una arquitectura aplicativa no está completa por tan solo separar su funcionalidad en capas, la arquitectura va más allá de eso, el propósito de que una Aplicación Empresarial cuente con una arquitectura es que la aplicación sea disponible, escalable y mantenible, pero además de esto se pretende basar su desarrollo en componentes reutilizables.

La reusabilidad es un tema que no se tratará en este documento ya que la teoría de la reusabilidad es muy amplia y no es el objetivo de este trabajo, sin embargo la reusabilidad es un resultado de una arquitectura bien definida. Es por ello que en este capítulo se presentarán los componentes que conforman la arquitectura con la cual se pueden construir Aplicaciones Empresariales.

En este capítulo se describirán los componentes que intervienen en la creación de una aplicación empresarial, para ello se presentan en la figura 2.8 las capas del Patrón General con la propuesta de los componentes lógicos que las deben conformar.

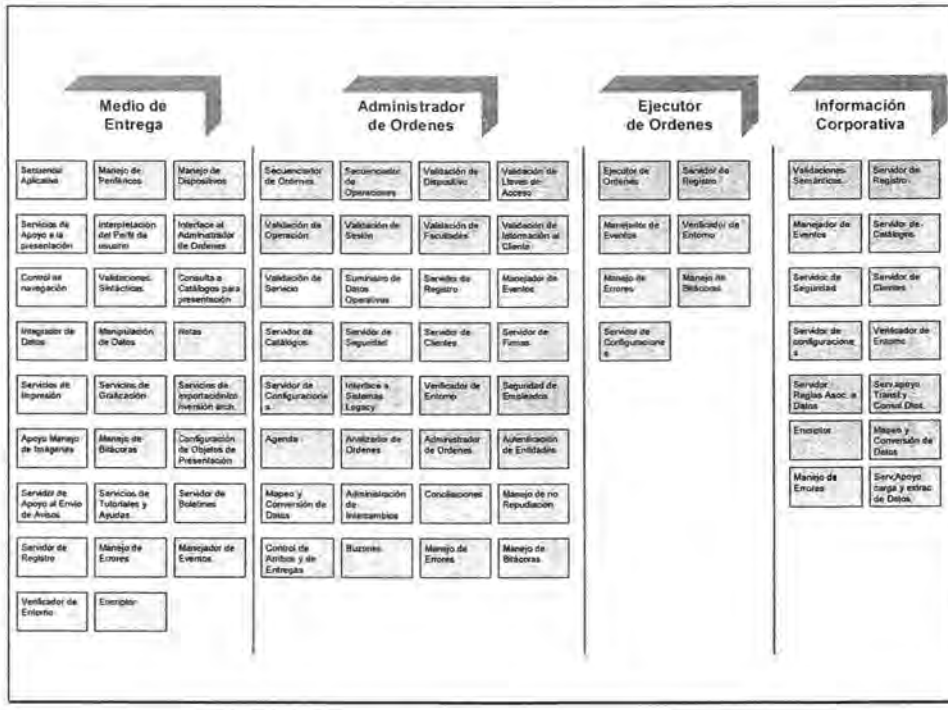

Figura 2.8 El Patrón General y sus componentes

El propósito de este capítulo es describir la funcionalidad de estos componentes y mostrar el por qué de la división en capas.

2.3.2 La capa del Medio de Entrega

En la tabla 2.3 se presentan los componentes asociados a la capa del Medio de Entrega y la funcionalidad que deben cubrir.

COMPONENTE	FUNCIONALIDAD
Secuencia Aplicativa	Recibir las peticiones del usuario y llamar al componente que resolverá total o parcialmente dicha petición.
Manejo de Periféricos	Entabla y controla la interacción con los diferentes periféricos asociados al Medio de Entrega.
Manejo de Dispositivos	Se encarga de la identificación del dispositivo por el cual se están recibiendo las peticiones del usuario, además identifica el formato de comunicación con el dispositivo de entrada.
Servicios de Apoyo a la presentación	Presenta encabezados, pies de página, contenido en general que personaliza la presentación.
Interpretación del Perfil de usuario	Identifica y administra el perfil del usuario con la finalidad de acotar y controlar la funcionalidad que el usuario puede realizar.
interfase al Administrador de Ordenes	Este es el conector hacia la capa del Administrador de Ordenes.
Control de navegación	Permite que el usuario navegue en la aplicación identificando el estado en que se encuentra.
Validaciones Sintácticas	Valida que la captura de información que hace el usuario se sintacticamente correcta.
Consulta a Catálogos para presentación	Presenta información de los catálogos.
Integrador de Datos	Integra la información recibida de la capa del Administrador de Ordenes
Manipulación de Datos	Complemento del Integrador de Datos, ordena los datos de acuerdo a las reglas de la presentación.
Notas	Información adicional asociada al funcionamiento del sistema, son tips que se presentan al usuario y pueden formar parte del encabezado o pie de página.
Servicios de Impresión	Interfase con los servicios de impresión, es controlada a través del Manejador de Periféricos.
Servicios de Graficación	Interfase con los servicios de graficación, es controlado a través del Manejador de Periféricos.
Servicios de exportación/conversión archivos.	Interfase con los servicios de exportación y conversión de archivos, es controlado a través del Manejador de Periféricos.
Apoyo Manejo de Imágenes	Servicio de mapeo y paginación de imágenes en el caso de que estas sean mayores al área de trabajo.
Manejo de Bitácoras	Interfase con la bitácora en donde se deja información para seguimiento.
Configuración de Objetos de	Permite configurar los objetos de esta capa para personalizarlos de acuerdo a la aplicación, esto sin necesidad de recompilarlos.

Presentación	
Servidor de Apoyo al Envio de Avisos	Permite generar notificaciones que no necesitan de respuesta inmediata.
Servicios de Tutoriales y Ayudas	Ayuda al usuario.
Servidor de Boletines	Recibe y presenta información de boletines.
Servidor de Registro	Registra los diferentes eventos generados y registra en el medio correspondiente.
Manejo de Errores	Atrapa los errores para su manipulación.
Manejador de Eventos	Captura, filtra y redirecciona los eventos que necesiten ser atendidos.
Verificador de Entorno	Comprueba el entorno aplicativo para procesar o no una petición.
Encriptor	Encripta la información de acuerdo al protocolo de comunicación.

Tabla 2.3 Componentes del Medio de Entrega

2.3.3 La capa de Administración de Ordenes

En la tabla 2.4 se presentan los componentes asociados a la capa del Administrador de Órdenes y la funcionalidad que deben cubrir.

COMPONENTE	FUNCIONALIDAD
Secuenciador de Ordenes	Preprocesa, procesa y postprocesa la operaciones que deban ser atendidas en batch.
Secuenciador de Operaciones	Preprocesa, procesa y postprocesa las operaciones que deban ser atendidas en línea.
Validación de Dispositivo	Valida que el dispositivo que entrega el requerimiento sea valido.
Validación de Llaves de Acceso	Valida las llaves de acceso y permite la firma del usuario.
Validación de Operación	Valida que la operación a ejecutar cumpla con las reglas establecidas por el negocio.
Validación de Sesión	Verifica que la sesión sobre la cual se ampara una operación este vigente.
Validación de Facultades	Verifica que el usuario que esta firmado pueda ejecutar el requerimiento establecido.
Validación de Información al Cliente	Verifica que la información proporcionada por el cliente corresponda a su perfil.
Validación de Servicio	Verifica que el servicio solicitado este disponible antes de enviar el requerimiento.
Suministro de Datos Operativos	Obtiene los datos necesarios para completar una operación.
Servidor de Registro	Registra los diferentes eventos generados y registra en el medio correspondiente.

Manejador de Eventos	Responde a los diferentes eventos generados en la aplicación.
Servidor de Catálogos	Contiene y proporciona información contenida en los catálogos.
Servidor de Seguridad	Contiene y proporciona información de seguridad.
Servidor de Clientes	Contiene y proporciona información de los clientes.
Servidor de Firmas	Contiene y proporciona información de la firma única del cliente.
Servidor de Configuraciones	Contiene y proporciona información de configuración aplicativa para esta capa.
interfase a Sistemas Legacy	Conector con host.
Verificador de Entorno	Verifica que el entorno aplicativo sea el correcto para poder atender la operación recibida.
Seguridad de Empleados	Permite el acceso y firma de los empleados.
Agenda	Aplicación de agenda para el cliente o usuario.
Analizador de Ordenes	Verifica que la orden recibida cumpla con los estándares y formatos establecidos.
Administrador de Ordenes	Controla los componentes que se encargan de resolver las órdenes.
Autenticación de Entidades	Autentifica la entidad a la cual pertenece el empleado o cliente.
Mapeo y Conversión de Datos	Traductor de formatos.
Administración de Intercambios	Controla el intercambio de información con otras capas.
Conciliaciones	Puntea las operaciones contables.
Manejo de no Repudiación	Controla que el servicio no sea negado a quien tiene derecho a recibirlo.
Control de Arribos y de Entregas	Controlador de la transferencia de archivos.
Buzones	Buzones para entregar información de la transferencia de archivos.
Manejo de Errores	Recibe la notificación de errores y los procesa.
Manejo de Bitácoras	Recibe la notificación de eventos y errores para registrarlos en la bitácora.

Tabla 2.4 Componentes del Administrador de Ordenes

2.3.4 La capa del Ejecutor de Ordenes

En la tabla 2.5 se presentan los componentes asociados a la capa del Ejecutor de Órdenes y la funcionalidad que deben cubrir.

COMPONENTE	FUNCIONALIDAD
Ejecutor de Ordenes	Ejecuta una orden.
Servidor de Registro	Registra los diferentes eventos generados.

Manejador de Eventos	Captura, filtra y redirecciona los eventos que necesiten ser atendidos.
Verificador de Entorno	Verifica que el entorno aplicativo sea el correcto para poder atender la operación recibida.
Manejo de Errores	Recibe la notificación de errores y los procesa.
Manejo de Bitácoras	Recibe la notificación de eventos y errores para registrarlos en la bitácora.
Servidor de Configuraciones	Contiene y proporciona información de configuración aplicativa para esta capa.

Tabla 2.5 Componentes del Ejecutor de Ordenes

2.3.5 La capa de Información Corporativa

En la tabla 2.6 se presentan los componentes asociados a la capa de Información Corporativa y la funcionalidad que deben cubrir.

COMPONENTE	FUNCIONALIDAD
Validaciones Semánticas	Repositorio con la información para resolver las validaciones semánticas.
Servidor de Registro	Repositorio de registro.
Manejador de Eventos	Contiene el repositorio para almacenar los eventos generados.
Servidor de Catálogos	Contiene la información para resolver las peticiones al servidor de catálogos.
Servidor de Seguridad	Contiene la información para resolver las peticiones de seguridad.
Servidor de Clientes	Contiene la información para resolver las peticiones de acceso a clientes
Servidor de configuraciones	Contiene la información para resolver las peticiones de acceso a configuraciones.
Verificador de Entorno	Contiene la información para resolver las peticiones de verificación de entorno.
Servidor de Reglas Asociadas a Datos	Contiene la información de las reglas que aplican a los datos.
Servidor de apoyo, transferencia y Consolidación de Datos.	Apoyo para el punteo de información.
Encriptor	Encripta la información.
Mapeo y Conversión de Datos	Traductor de formatos.
Manejo de Errores	Recibe la notificación de errores y los procesa.
Servidor de apoyo, carga y extracción de Datos.	Carga y extrae información masiva.

Tabla 2.6 Componentes de la Información Corporativa

La finalidad de contar con estos componentes es cubrir la funcionalidad necesaria en cada una de las capas con el propósito de construir aplicaciones empresariales basadas en componentes reutilizables, esto para acelerar el desarrollo, minimizar los costos y tener un "Time to Market" menor que permita a las empresas cumplir con los nuevos retos y requerimientos en un mejor tiempo, logrando de esta forma no solo sobrevivir, sino también ser competitivas al poder atender con prontitud las nuevas necesidades del mercado.

Todo lo anterior puede lograrse de dos formas, la primera de ellas es invirtiendo mucho dinero y recursos para construir nuevas aplicaciones completas e independientes por cada requerimiento nuevo que surja, y la segunda, construir nuevas aplicaciones basadas en capas y componentes reutilizables con la funcionalidad acotada y plenamente identificada para poder atender a más de una aplicación, esto con el ahorro en tiempo, dinero y recursos que resulta en una enorme ventaja tecnológica que facilitará a la empresa ser líder en su sector.

Sin embargo, el contar con estos componentes no es suficiente ya que no es posible poder construir aplicaciones basadas solo en piezas que cubren cierta funcionalidad, falta aún otro elemento para poder tener aplicaciones robustas, escalables y con un alto nivel de disponibilidad, dicho elemento es la arquitectura.

CAPÍTULO 3
CONSTRUCCIÓN DE LA ARQUITECTURA

En este capítulo se presenta la segunda parte de la propuesta realizada en este trabajo, la primera parte consistió en la presentación de los componentes arquitectónicos que realizarán todas las funciones comunes necesarias para una aplicación empresarial, mientras que la segunda parte consistirá en la construcción e implementación de la arquitectura sobre la cual estos componentes actúan.

La arquitectura de una aplicación es muy importante no solamente para proveer a los componentes de una plataforma sobre la cual desempeñarse, si no también porque la arquitectura proporcionará a la aplicación la posibilidad de obtener un buen rendimiento al proporcionarle las capas arquitectónicas necesarias mediante la implementación de plataformas aplicativas en donde los componentes de cada una de las capas se desempeñan.

3.1 El medio de entrega

Los primeros elementos a considerar son los dispositivos clientes, con estos se representa al(los) usuario de(los) sistema(s) a crear o el destino en donde se usarán los datos, este destino o usuario puede ser una persona u otra aplicación. En este paso se debe incluir en la arquitectura un elemento que nos represente el dispositivo por el cual nuestro usuario estará interactuando, el dispositivo claramente dependerá del requerimiento a cubrir, por ejemplo, en el caso de una Aplicación Web el dispositivo cliente será una Computadora Personal con un Web Browser, pero para algún otro caso el dispositivo cliente puede ser una Palm, una Computadora de Bolsillo, un Web Server, un celular, otra aplicación, un controlador, etc.

En esta propuesta a los clientes se les ubica en el extremo izquierdo de la arquitectura ya que generalmente son quienes generan el primer evento y son los solicitantes de la información. Es necesario así mismo considerar el número de clientes que se estarán conectando concurrentemente a la aplicación así como la variedad de estos (versiones de sistema operativo, de Web Server, velocidades de procesamiento, medios de comunicación, etc.) con la finalidad de poder representar en la arquitectura al dispositivo o dispositivos necesarios y tener una apreciación de la posible cantidad de clientes.

Para determinar el número de clientes debe tomarse en cuenta:

1.- Si la aplicación es Web y saldrá por Internet.
2.- Si la aplicación es Web y estará en la Intranet.
3.- Si la aplicación tendrá un cliente pesado (Visual Basic, programa Java, programa en C, etc.).
4.- Si la aplicación tendrá como usuario a otra aplicación.

Consideraciones importantes:

1.- De acuerdo al tipo de aplicación, se podrán tener uno o más dispositivos clientes.
2.- El color gris se utiliza para representar aplicaciones comerciales.
3.- El color verde se utiliza para representar componentes internos ya disponibles.
4.- Dentro de cada cliente deben ponerse los componentes necesarios para que el usuario pueda manipular y conectarse a la aplicación.
5.- Los componentes que interactúen entre si deben estar ligados.
6.- La representación debe incluir el Sistema Operativo (esquina superior derecha), componentes (cuerpo) y nombre del dispositivo, máquina o entidad de uso (base).

Ejemplo

El cliente de una la aplicación Internet comúnmente usará un Browser, pero la aplicación puede ser acesada desde Internet, desde Intranet y desde una aplicación VB ya existente para generar estadísticas sobre el uso del sistema. Por lo anterior para este caso se debe notar que contamos con 2 diferentes tipos de dispositivos con los cuales los clientes interactúan: Un browser (internet / Intranet) y una aplicación Visual Basic.

Con la información anterior la primera aproximación de la arquitectura se representa en la figura 3.1.

3.2 Presentación en la capa de Medios de Entrega

Lógicamente dentro de la capa de Medios de Entrega se ubican también los componentes del Apoyo a la presentación, por esto los elementos que debemos considerar después de nuestros clientes son los dispositivos a los cuales nuestros clientes se conectarán, generalmente estos dispositivos son los WebServers, ya que estos dispositivos permiten la conectividad con el cliente y sirven como *Concentradores* de conexiones. En este paso se debe incluir en la arquitectura un elemento que nos represente el dispositivo con el cual se estarán conectando los clientes.

A los WebServers y/o Concentradores se les ubica a la derecha de los clientes, después de pasar por la Internet, Intranet o LAN. Pueden tener presentación o no y contienen todos los elementos que apoyan a la presentación, es decir, componentes que no tienen mayor lógica que aquella que compete a la capa de la presentación.

Para determinar el número de WebServers y/o Concentradores debe tomarse en cuenta:

1.- Si es una aplicación Internet, el WebServer debe representarse dentro de la Zona Desmilitarizada (Firewall y PIX).
2.- Si la aplicación es Web y saldrá por Internet.
3.- Si la aplicación es Web y estará en la Intranet.
4.- Si la aplicación es Cliente / servidor.

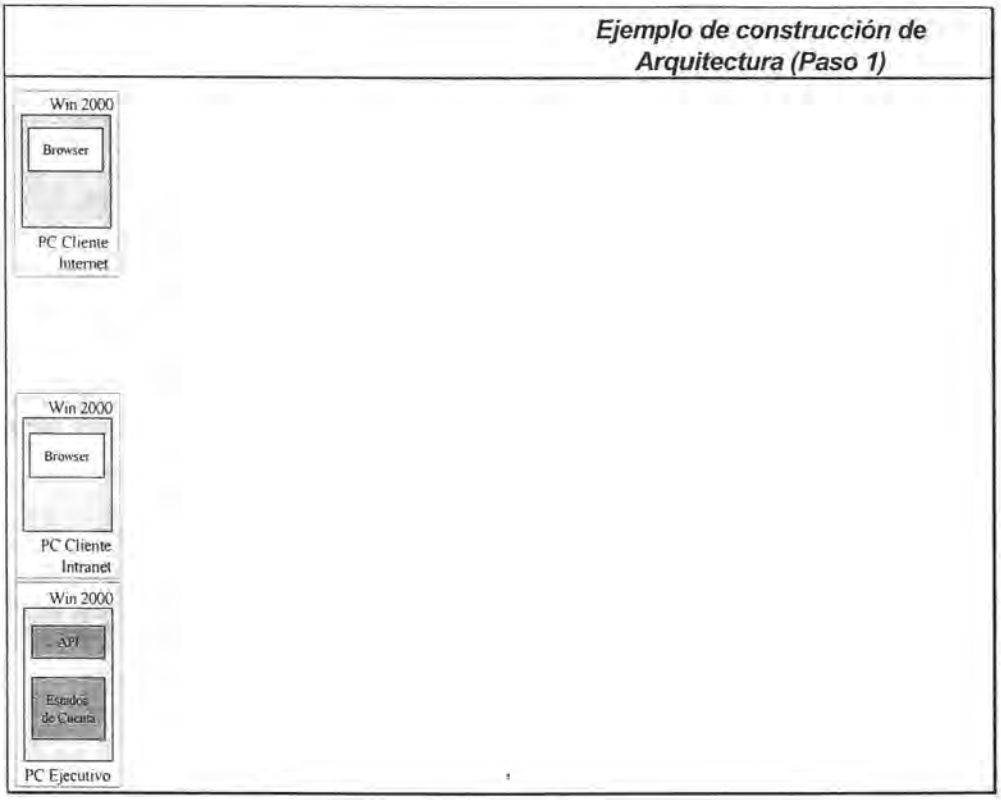

Figura 3.1 El medio de entrega

Consideraciones importantes:

1.- Internet Information Server (IIS) se usa comúnmente para aplicaciones internas que no salen a Internet y que no tienen que ver directamente con el negocio.
2.- iPlanet se usa comúnmente para aplicaciones que salen a Internet y que tienen que ver directamente con el negocio.
3.- En los WebServers deben incluirse los elementos de presentación (páginas html, imágenes, vbx, etc.) que no contengan lógica de negocio.
4.- En el caso de no necesitar un WebServer el balanceador se utiliza para concentrar las conexiones hacia los servidores intermedios.

Ejemplo

Como la aplicación puede ser acesada desde Internet será necesario contar con un WebServer que se encuentre en la zona desmilitarizada, sin embargo, la aplicación también será usada desde Internet por lo que se debe considerar el uso de un WebServer para este tipo de clientes. La aplicación Visual Basic debe usar un application program

interfase (API) para la conectividad hacia el monitor de transacciones (en este caso Tuxedo), por ello en la arquitectura se utiliza un concentrador que se encuentra en este caso en la misma máquina donde esta el WebServer iPlanet.

Con la información anterior la segunda aproximación de la arquitectura se representa en la figura 3.2.

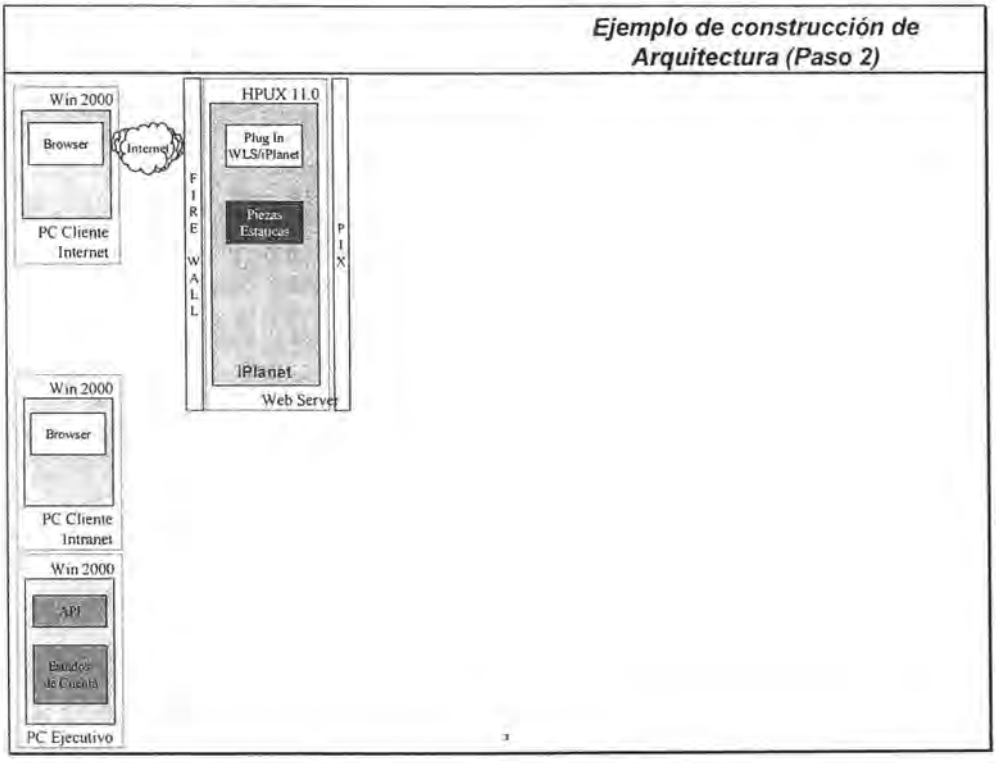

Figura 3.2 El Apoyo a la presentación (medio de entrega)

Ya con los primeros elementos representados se procede a representar en el diagrama las conexiones. En este paso se debe incluir el flujo de información poniendo sentido en las flechas que unen los diferentes elementos.

Consideraciones importantes:

1.- La comunicación entre elementos de la misma naturaleza dentro de una misma máquina se representa con flechas simples.
2.- La comunicación entre elementos de diferente naturaleza dentro o fuera de una misma máquina se representa con flechas con indicación del protocolo usado.

Ejemplo

Se debe enlazar con flecha simple la aplicación de Estados de Cuenta y el API, esto es así porque la comunicación es interna y sin necesidad de usar ningún protocolo especial, lo mismo ocurre entre las piezas estáticas del webserver y el plugin del servidor aplicativo (Weblogic Server). Por otra parte, los clientes browser se comunican con el WebServer haciendo uso de http, en este punto es importante mencionar que se debe hacer diferencia entre HTTPS (Internet) y HTTPS (Intranet), por ello la etiqueta que indica el protocolo así lo representa.

Con la información anterior la segunda aproximación de la arquitectura se representa en la figura 3.3.

3.3 Diseño de la capa de Integración de Funcionalidad de Negocio

Ahora debemos representar el servidor aplicativo a usar o la capa en donde residirá la lógica aplicativa, generalmente esta se encontrará dentro del servidor aplicativo que para nuestro caso es WebLogic Server. En este servidor intermedio se coloca la secuencia aplicativa (flujo de pantallas), los componentes de la aplicación, los componentes de acceso a los datos (que por si mismos son una capa lógica) y si la aplicación así lo requiere los componentes que permitan la comunicación al Monitor de Transacciones. Se debe incluir también las bases de datos a usar que se encuentren en la parte intermedia y en general cualquier componente o software que se ubique en este punto.

Consideraciones importantes:

1.- En este punto (servidor intermedio) deben incluirse los elementos que contengan lógica aplicativa (Servlets y JSP por ejemplo).
2.- Así mismo debe incluirse también los elementos que acceden a los datos.
3.- Si se usa otro tipo de software o componentes que residan en este servidor intermedio deben representarse.
4.- En el caso de Tuxedo, generalmente solo se publican servicios que físicamente se encuentran en otra máquina.
5.- Por claridad, es conveniente separar con líneas los diferentes ambientes dentro de este servidor intermedio (WebLogic, Sybase, Tuxedo, etc.).

Ejemplo

Nuestra aplicación Web se compondrá de elementos que contengan una secuencia aplicativa en especifico que a su vez se auxilie de otros componentes para acceder a los datos que se encuentran contenidos en Sybase, por otra parte, nuestra aplicación Web también debe comunicarse con el Monitor de Transacciones (Tuxedo) para dar y obtener información, por ello se representa el conector hacia la siguiente capa, en este caso y dado el uso de WebLogic Server se usará WebLogic Tuxedo Connector (WTC) y el

conector que nos ligará hacia la siguiente capa. Así mismo se representan en Tuxedo los servicios a usar mediante el símbolo para representar a los servicios Tuxedo.

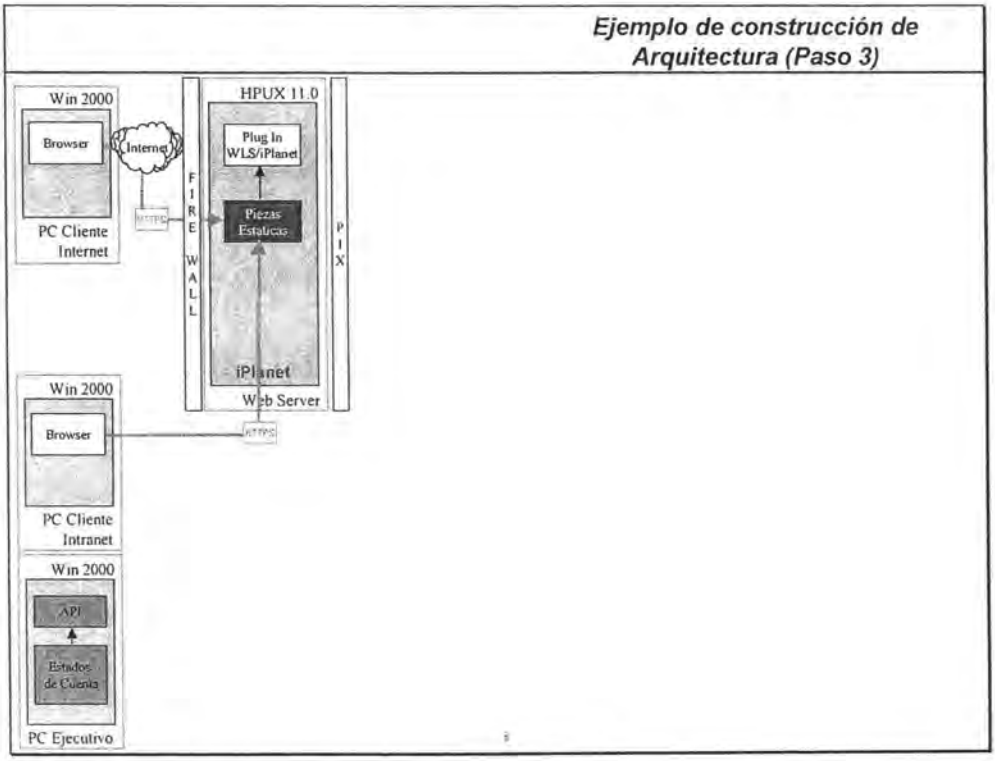

Figura 3.3 El medio de entrega y apoyo a la presentación

Con la información anterior la cuarta aproximación de la arquitectura se representa en la figura 3.4.

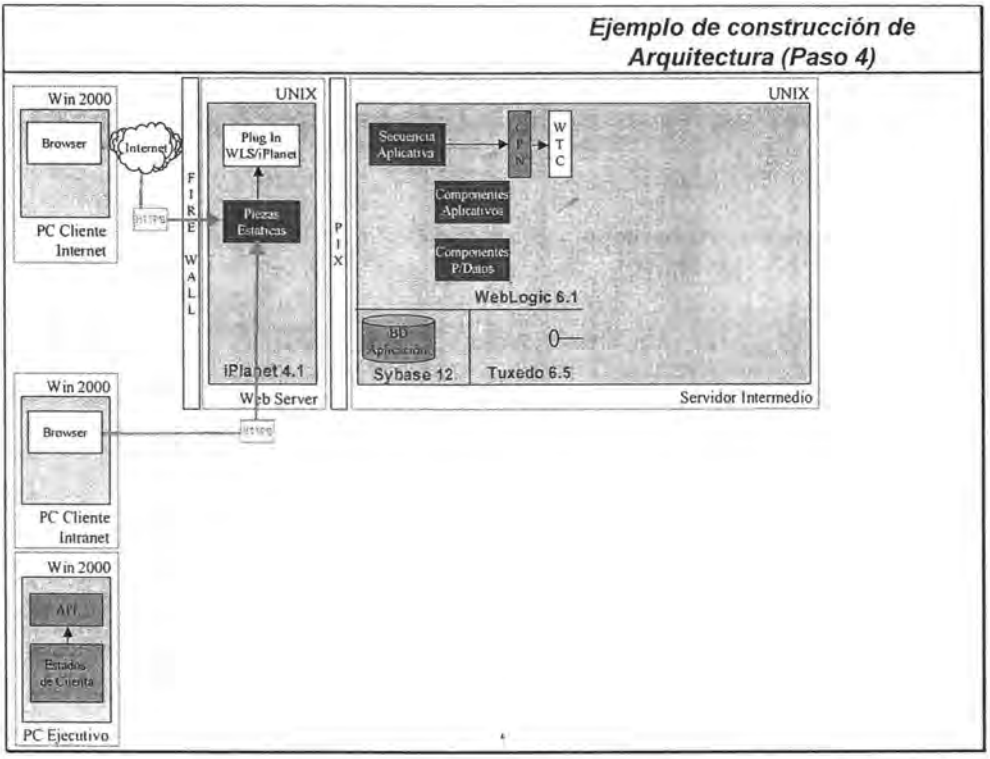

Figura 3.4 Componentes y elementos de la capa del Administrador de Ordenes.

3.4 Ejecutor de Órdenes

Ya con los elementos del servidor intermedio se procede a pintar las conexiones. En este paso también se debe incluir el flujo de información poniendo sentido en las flechas que unen los diferentes elementos.

Consideraciones importantes:

1.- La comunicación entre elementos de la misma naturaleza dentro de una misma máquina se representa con flechas simples.
2.- La comunicación entre elementos de diferente naturaleza dentro o fuera de una misma máquina se representa con flechas con indicación del protocolo usado.

Ejemplo

Se debe enlazar con flecha simple todos los componentes que están dentro de WebLogic, sin embargo, el acceso a los datos (Sybase) debe hacerse utilizando una conexión JDBC, por ello se representa esa conectividad auxiliándonos de una etiqueta que nos indica el uso de dicho protocolo. Lo mismo sucede para la comunicación entre la aplicación Visual Basic y el concentrador Tuxedo.

Con la información anterior la quinta aproximación de la arquitectura se representa en la figura 3.5.

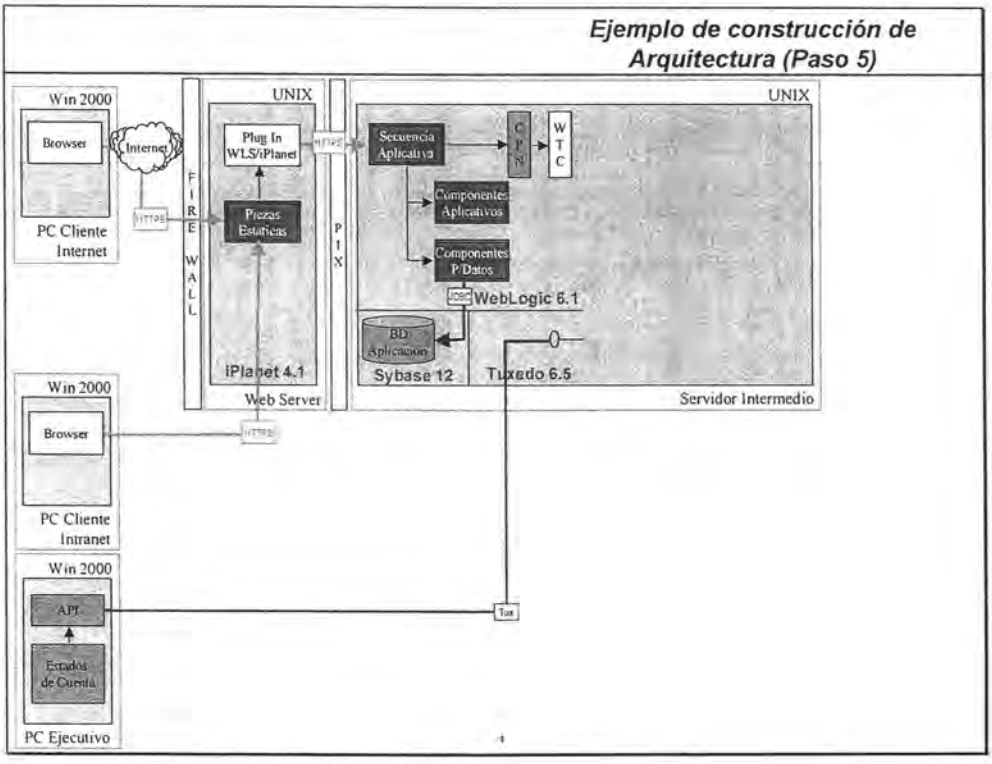

Figura 3.5 Componentes y elementos de la capa del Administrador de Ordenes relacionados.

Es en este momento en donde la arquitectura toma un papel muy importante para lograr la reusabilidad, y es que es en este punto donde se proceden a representar los componentes del Monitor de Transacciones que serán utilizados. El monitor de transacciones junto con el servidor aplicativo son los dispositivos que contendrán la mayoría de los componentes de los cuales se conforme un sistema. En este paso se deben incluir piezas que permiten tanto la entrada a Monitor de Transacciones (Preproceso), como el acceso a sistemas centrales (traductor). Además, la aplicación

ejemplo necesita consultar información de Catálogos. Por otra parte se representan el servidor de log para manejo de eventos y registro de log mediante el Resolutor de Operaciones y es en este último elemento en donde residirá la lógica de negocio de la aplicación. Nótese que tanto el Preproceso como la librería de catálogos cuentan ya con una flecha que los liga a una base de datos, esto es así porque dichas bases son parte de la misma pieza.

Consideraciones importantes:

1.- La entrada a Monitor de Transacciones siempre será por el Preproceso.
2.- Si se desea acceder a sistemas centrales entonces debe usarse el componente traductor.
3.- El servidor de log se incluye siempre que se use Monitor de Transacciones.
4.- En el Resolutor de Operaciones residirá la lógica de negocio que resolverá la aplicación.

Ejemplo

La aplicación usará el Resolutor de Operaciones porque en esta pieza es donde se resolverá la lógica de negocio de la aplicación. Por otra parte se incluye la librería de catálogos para que el sistema pueda obtener la información que necesita de los catálogos. Como el sistema también necesita información de sistemas centrales se incluye el componente traductor el cual será configurado para poder establecer la comunicación adecuada hacia los sistemas heredados (legacy). El servidor de log se incluye también como parte de la infraestructura de Monitor de Transacciones.

Con la información anterior la sexta aproximación de la arquitectura se representa en la figura 3.6.

Ahora se debe proceder a conectar los dispositivos "clientes" del Monitor de Transacciones, como sus elementos internos. En este paso al igual que con los anteriores se procede a pintar las conexiones que en su gran mayoría serán flechas simples. También se debe incluir el flujo de información poniendo sentido en las flechas que unen los diferentes elementos.

Consideraciones importantes:

1.- La comunicación entre elementos de la misma naturaleza dentro de una misma máquina se representa con flechas simples.
2.- La comunicación entre elementos de diferente naturaleza dentro o fuera de una misma máquina se representa con flechas con indicación del protocolo usado.
3.- La comunicación aceptada entre WebLogic Server y Monitor de Transacciones (Tuxedo) se efectúa mediante el uso de dominios establecidos entre Weblogic Tuxedo Connector y Tuxedo.

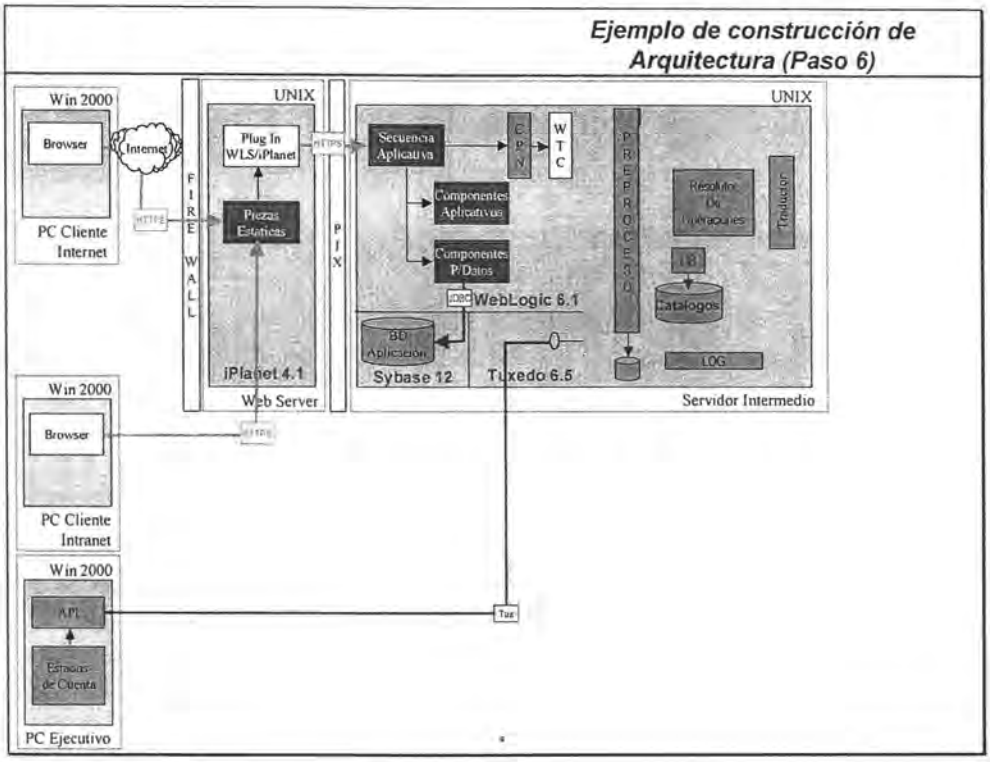

Figura 3.6 El Ejecutor de Ordenes

Ejemplo

Se debe enlazar con flecha simple todos los componentes que están dentro de Monitor de Transacciones, recordando que el Resolutor de Operaciones es la pieza central en esta capa ya que es la que resuelve la operación (o conjunto de transacciones). La comunicación desde el Servidor Aplicativo (WebLogic Server) así como la del API (conector) usarán el protocolo propietario del monitor de Transacciones (en este caso Tuxedo).

En este punto es importante mencionar que cada operación será validada en la cada capa antes de llegar al Resolutor de Operaciones donde además se efectuará un preproceso a la operación y un postproceso con la finalidad de que cada respuesta este completamente realizada o en caso de rechazo efectuar las reversas necesarias.

Con la información anterior la séptima aproximación de la arquitectura se representa en la figura 3.7.

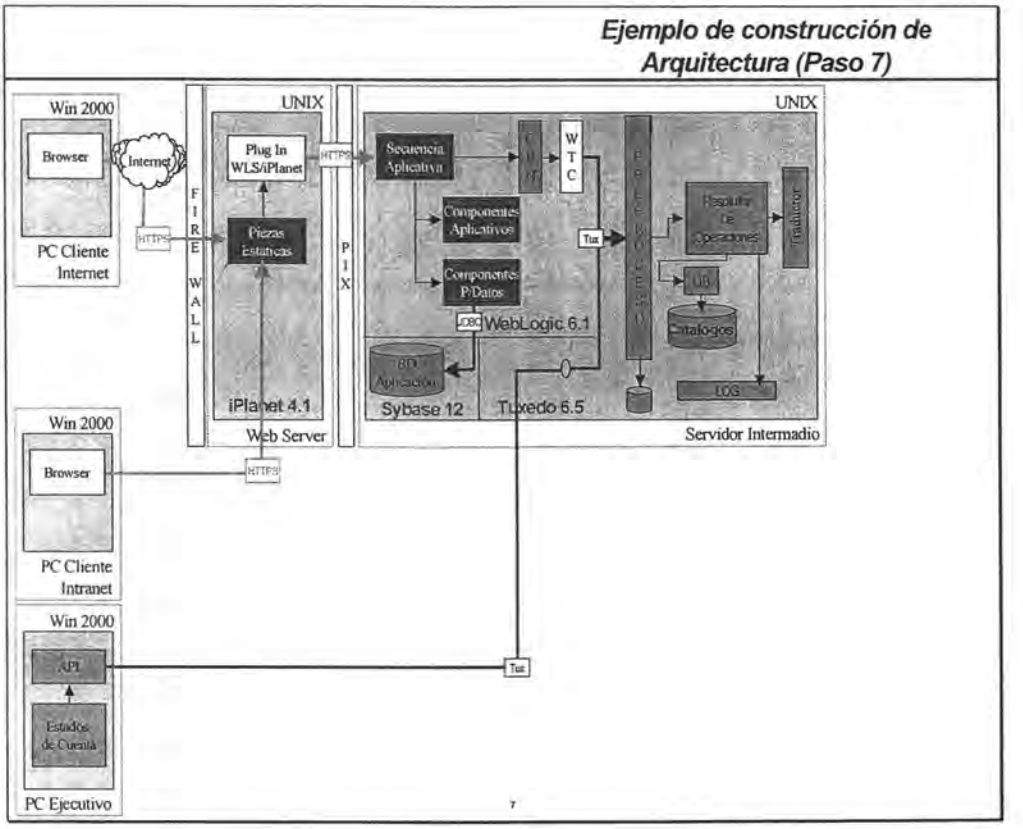

Figura 3.7 El Ejecutor de Ordenes con el preproceso

3.5 La Información Corporativa

Ahora es el momento de representar en la arquitectura los sistemas heredados (legacy) a los cuales queremos llegar o de los cuales necesitamos información, además también este es el momento de representar cualquier repositorio de datos al cual debamos acceder, estos repositorios y sistemas legacy tendrán sus componentes de acceso a datos, en esta capa lo importante es proteger al dato encapsulándolo con servicios o funcionalidad que modifique su estado, pero evitando que el dato sea modificado por funcionalidad externa. Hay que hacer notar que la arquitectura se basa en capas lógicas y no en capas físicas, esto quiere decir que puede haber en esta última capa otro tipo de elementos diferentes a los sistemas legacy y es por ello que se incluyen bases de datos. En este paso se incluyen todos los sistemas a los cuales deseamos conectarnos, además deben representarse las piezas que nos darán acceso a los datos (servicios de acceso a datos).

Consideraciones importantes:

1.- El resolutor de Operaciones puede comunicarse directamente a servicios de acceso a datos (servicios Tuxedo).
2.- El traductor se utiliza para poder establecer comunicación con aplicaciones o bases de datos en donde la comunicación no sea necesariamente TCP/IP.

Ejemplo

Nuestro ejemplo debe conectarse a aplicaciones que están en Tandem y aplicaciones que están en Unisys, por ello se representan esas máquinas y los elementos que utilizaremos de ellas (Base de Datos 1, Base de Datos 2 y Base de Datos 3) para Unisys y la BD 4 para Tandem.

Con la información anterior la octava aproximación de la arquitectura se representa en la figura 3.8.

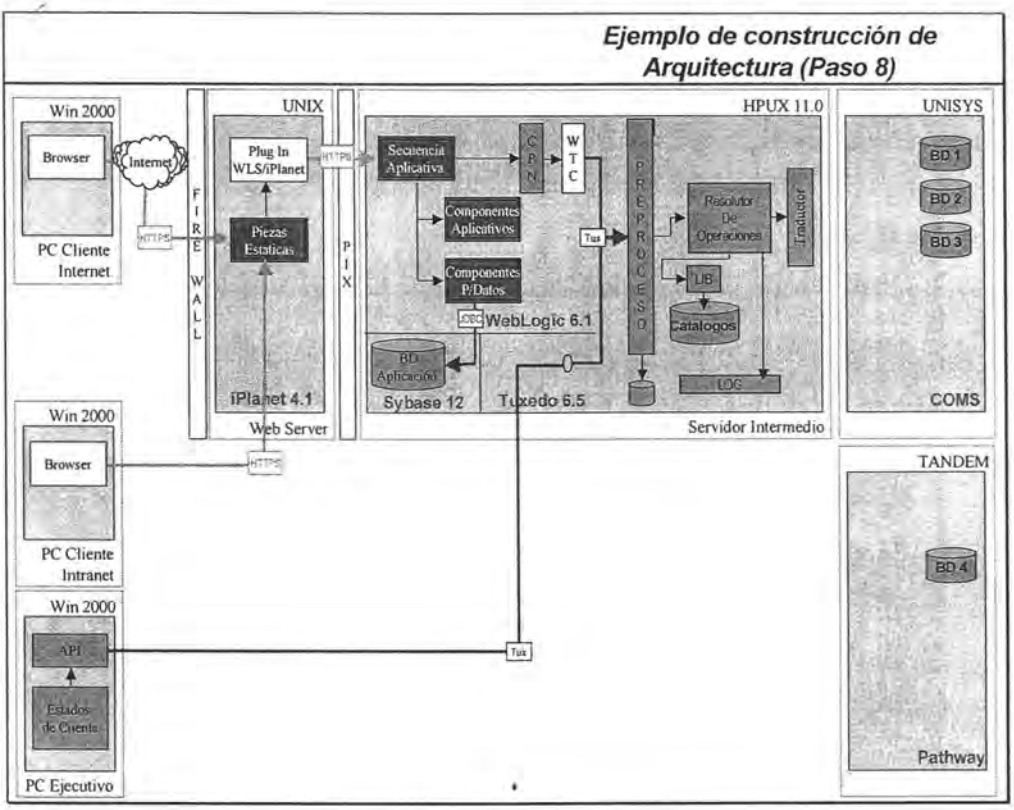

Figura 3.8 La Información Corporativa

Ahora que ya se cuenta con todos los elementos necesarios se debe proceder a conectar los elementos faltantes, la mayoría de las flechas o conexiones se representarán mediante el uso de flechas simples. Nótese que el Resolutor de Operaciones se comunica directamente a un servicio Tuxedo, pero que necesita del traductor para acceder a sistemas que no pueden usar TCP/IP.

Consideraciones importantes:

1.- La comunicación entre elementos de la misma naturaleza dentro de una misma máquina se representa con flechas simples.
2.- La comunicación entre elementos de diferente naturaleza dentro o fuera de una misma máquina se representa con flechas con indicación del protocolo usado.
3.- La comunicación entre Monitor de Transacciones y los sistemas legacy se efectúa usando el traductor para el caso de aplicaciones sin TCP/IP o usando Tuxedo en el caso de llamar a servicios del Monitor de Transacciones.

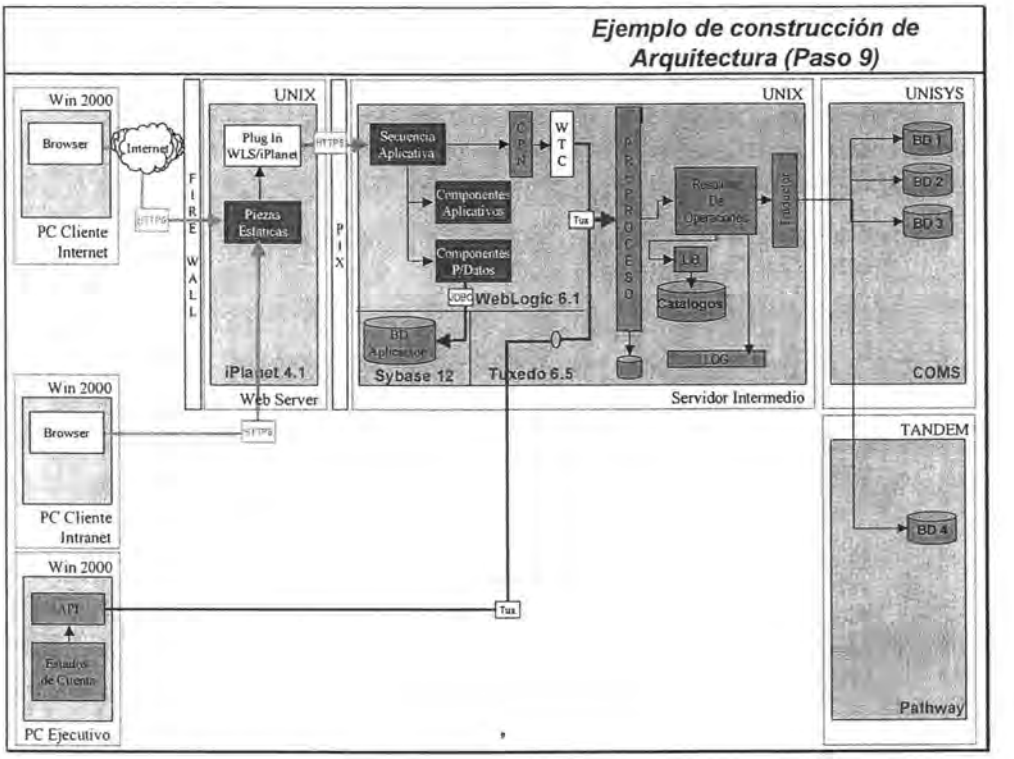

Figura 3.9 La Arquitectura Terminada

Ejemplo

Se debe enlazar con flecha simple todos los elementos a los cuales se comunica tanto al traductor como al Resolutor de Operaciones.

Con la información anterior completamos la arquitectura la cual que representada con todas las capas en la figura 3.8.

Los pasos que se han mostrado llevan a completar la arquitectura aplicativa de una aplicación empresarial. Los pasos anteriores sirven de manera general para el diseño de una solución típica, es claro que cada aplicación tendrá miles de variantes, sin embargo lo importante es crear un esqueleto basado en componentes reusables y capas arquitectónicas bien definidas para poder construir aplicaciones en bloques, reduciendo de esta manera costos y tiempo de desarrollo, dando a la empresa una ventaja competitiva.

CAPÍTULO 4

LA IMPLEMENTACIÓN

Ahora que tenemos a la aplicación dividida en capas y los componentes que la conforman ya identificados es necesario implementar el proyecto, esto no es otra cosa mas que aterrizar las ideas plasmadas y crear el sistema, es decir llevarlo a cabo.

4.1 La plataforma operativa a usar

Esta es una decisión difícil de tomar, básicamente nos enfrentamos a decidir entre 2 tecnologías: la propietaria en donde el principal representante es Microsoft y la abierta que actualmente esta representada por Java y UNIX. En este trabajo no se discutirá cual es la mejor plataforma a usar para implementar este tipo de aplicaciones, sin embargo lo que si pretende ser es una guía para elegir la que mejor se adapte al presupuesto y necesidades de la empresa.

La tecnología propietaria generalmente es cara, pero tiene una ventaja: Garantia, y generalmente viene acompañada de un grupo de expertos que con el afán de vender implementarán una primera versión de la aplicación y realizarán la transferencia de la tecnología a la empresa. Otra ventaja es la que generalmente este tipo de tecnología es dominada y explotada por aliados de negocio (partners) que ofrecen servicios de consultaría, de esta manera la empresa que utilice la tecnología propietaria puede estar segura de que siempre contará con alguien que pueda darle soporte técnico en caso necesario.

Pero la tecnología propietaria tiene sus carencias, por ejemplo: su certificación se reduce a los aliados de negocio de la compañía dueña de la tecnología, esto quiere decir que la empresa podría ver afectada su inversión y sus gastos elevados al tener que adquirir equipo especial para soportar a la plataforma elegida. También debe considerarse el hecho de que al depender de una tecnología propietaria se depende del proveedor, y si el proveedor desaparece la inversión tecnológica puede venirse abajo.

Por otra parte existe la posibilidad de seleccionar la tecnología abierta, este tipo de tecnología es barata ya que se encuentra disponible libremente y generalmente el código también es accesible, puede también encontrarse muy fácilmente información técnica y hasta código libre, puede correr sobre casi cualquier hardware y no se depende de ningún proveedor.

Sin embargo también tiene sus desventajas, primeramente no se tiene garantía, no se tiene el control de versiones e incluso pueden existir fallas de sistema no conocidas que nadie atienda, y se corre el riesgo de que ante una falla critica no se tenga el nivel de soporte técnico adecuado. Además al ser productos públicos estos pueden desaparecer del mercado, lo que puede dejar en desventaja a la empresa que utilice dicha tecnología

al no poder actualizar su software para ajustarse a las nuevas necesidades. En la tabla 4.1 se observan algunas ventajas y desventajas de estas tecnologías.

Tecnología	Ventajas	Desventajas
Propietaria	Garantía	Costo alto
	Soporte técnico	Dependencia de proveedor
	Consultaría	Certificación de hardware
	Fuentes de información	
Abierta	Barata	Sin garantía
	Fuentes de información	Sin consultaría
	Independencia de proveedor	Soporte sin compromiso

Tabla 4.1 Ventajas y desventajas de las tecnologías propietaria y abierta

Ante esta situación la decisión de usar cierto tipo de tecnología deberá basarse en la criticidad de la aplicación a construir, igualmente deberá considerarse el volumen de transacciones esperado así como la capacidad de procesamiento disponible, el presupuesto para el sistema y la infraestructura instalada sin olvidar el tiempo estimado de vida del sistema.

En este trabajo se ejemplifica con tecnología abierta como lo es Java que es soportada comercialmente por empresas como Sun Microsytems, iPlanet y BEA Systems, mientras que la base de datos será relacional.

4.2 La seguridad

La seguridad es un aspecto sumamente importante en cualquier sistema, sin embargo en un sistema distribuido que además esta expuesto a posibles intrusos no solo de la institución si no de externos a ella cuidar la privacidad de la información es una cuestión vital. En seguridad de la información se dice que "la seguridad es tan fuerte como su punto más débil", con esto se pretende explicar que no es suficiente contar con los más avanzados sistemas de autenticación ya que de nada serviría identificar bien a los usuarios del sistema y tener registro de todas las actividades que realicen en él si internamente se descuida el acceso a la base de datos en donde se encuentra la información vital. Esto quiere decir que tan importante es saber quien usa nuestro sistema como protegerlo de ataques internos o externos.

4.2.1 Secure Sockets Layer

La seguridad de la información no se basa únicamente en proteger la confidencialidad de los datos, también es necesaria la privacidad de estos y la recuperación en caso de contingencia.

La forma más común de proteger la privacidad de los datos es encriptando la información, para ello el método más común para es el Secure Sockets Layer (SSL), con este método

se protege al dato de ser visto por un tercero, el objetivo es cifrar la información para que en manos de un agente externo al sistema este no pueda hacer nada con la información, para ello se tienen diferentes métodos de cifrado o encripción en el mercado.

Sin embargo, Independientemente del método usado para cifrar la información es posible encontrar en Internet y de manera gratuita los algoritmos para cifrar y descifrar información, es posible encontrar algoritmos para cifrado a 40, 56 y 128 bits, lo importante no es el algoritmo usado ya que la mayoría ofrecen un buen nivel de cifrado, sin embargo lo que realmente es trascendente es el nivel de cifrado, es decir los bits con los que se realizan los cálculos para proteger la información, por ello es importante no usar cifrados menores a 128 bits ya que la mayoría de los algoritmos para romper la seguridad (decodificadores) también se encuentran públicamente en Internet y son efectivos si el cifrado es menor a 128. Si el cifrado es de 128 la tarea de decodificar la información sin el certificado raíz es realmente difícil, por lo que este nivel de encripción es el más usado cuando se trata de proteger información relevante, sin embargo es tan eficaz que el gobierno de los Estados Unidos de Norteamérica ha puesto restricciones para el uso de este tipo de cifrado, la razón es muy sencilla: a 128 bits de cifrado ellos mismos no podrían decodificar los mensajes que interceptan, por esto cuando una institución fuera de ese país necesita usar este nivel de encripción y para ello utiliza software desarrollado en dicho país es necesario pedir autorización a ese gobierno con la finalidad de poder obtener certificados y licencias para usar SSL a 128 bits.

Por lo tanto, la recomendación es usar siempre que sea posible cifrado a 128 bits, de lo contrario la privacidad de la información puede verse comprometida. Para ello basta con obtener certificados SSL a 128 bits para el WebServer así como para el Servidor aplicativo, comúnmente estos elementos cuentan ya con la posibilidad de realizar este tipo de encripción y comunicación de manera natural, tan solo es necesario configurarlos una ves adquiridos ya que los proveedores o desarrolladores del software incluyen en sus productos la posibilidad de utilizar estos mecanismos de seguridad fácilmente.

4.2.2 La seguridad interna

Hasta ahora se ha tratado el tema de la seguridad suponiendo que los únicos intrusos son los agentes externos a la institución, sin embargo esto no es lo más común ya que los robos e intrusiones electrónicas se hacen más desde adentro que desde afuera y en un buen número de oportunidades cuando la intrusión se realiza de forma externa es porque alguien de adentro esta involucrado. Por lo anterior es necesario proteger a la información también de los agentes internos, para ello se recomienda tomar las siguientes acciones preventivas:

Acción Preventiva	Detalle
Uso de passwords personales	Todos los usuarios que tengan acceso al sistema ya sea para mantenimiento del sistema o para interactuar con él deberán de contar con una clave personal e intransferible.
Rotación de passwords	Todos los passwords deben ser rotados periódicamente, esto limitará la ventana de tiempo en el que un password que hubiera sido interceptado pueda usarse.
Administración de passwords	Los passwords de sistemas comunes (no de usuarios) como los passwords de las consolas, monitores y súper usuarios

	deben ser administrados por el departamento de administración de la seguridad.
Transmisión cifrada	Siempre que un dato confidencial sea transmitido debe estar cifrado, debe entenderse por transmitido el hecho de que dicho dato salga de un entorno operativo para ubicarse en otro, es decir, que salga de la maquina en donde originalmente reside.
Almacenamiento cifrado	Todo dato confidencial debe almacenarse encriptado, esto protegerá al dato de ser visto por usuarios que tengan facultades para ver el contenido de las tablas de una base de datos, además esta práctica debe hacerse en todas las capas, no solo en la base de datos.
Facultamiento de usuarios	Todos los usuarios operativos deben tener facultades limitadas para usar los módulos de un sistema así como para visualizar los datos, esto evitará contar con súper usuarios no registrados.
Listas de control de acceso	Todos los componentes de un sistema deben contar con restricciones de uso, esto evitará que un sistema alterno haga uso de componentes no autorizados para él.
Validación de campos de captura	Todo campo de captura debe evitar la posibilidad de dar entrada a un "Caballo de Troya", por lo anterior todos los campos de captura deben validar tanto el tipo de dato que están recibiendo como la semántica de este.
Auditoria de bitácoras	El departamento de Seguridad de la Información debe verificar las bitácoras de manera periódica para validar que no se estén registrando en ella datos o información confidencial.
Verificación de código	El departamento de Seguridad de la Información debe verificar el código fuente cada vez que se realiza una liberación de versión para validar que no se muestren al usuario datos o información confidencial no autorizada.
Eliminar comentarios y debugs	El departamento de Seguridad de la Información debe verificar el código fuente cada ves que se realiza una liberación de versión para validar que no se tengan debugs encendidos que muestren al usuario datos o información confidencial no autorizada, de la misma manera para validar que no existan comentarios que ayuden a un tercero a comprometer la información.

Tabla 4.2 Ventajas y desventajas de las tecnologías propietaria y abierta

La lista presentada ayudará a evitar comprometer la información y de la misma manera reducirá la posibilidad de sufrir ataques para denegar el servicio. El lector puede pensar que son demasiados puntos a considerar, sin embargo toda inversión en seguridad nunca será demasiada si de ella depende la supervivencia de la empresa.

4.2.3 Los sistemas de back up

Dentro de la seguridad de la información también debe considerarse la posibilidad de perder la información por causas técnicas como parpadeos de luz, picos de voltaje o fallas

en los controladores entre muchas otras posibilidades como lo son también los desastres naturales tales como terremotos, inundaciones o incendios.

Ante este tipo de eventos la seguridad de la información también debe tomar medidas preventivas, es por ello que se recomienda tomar las acciones preventivas descritas en la tabla 4.2.

Acción Preventiva	Detalle
Resguardo del software	El software y procedimientos de instalación deben estar protegidos en diferentes ubicaciones geográficas, de esta manera se protege la inversión y siempre será posible recuperar tanto el código objeto como el código fuente de un sistema.
Respaldo de la información	Toda la información generada por un sistema debe ser respaldada diariamente en medios magnéticos para contar con la posibilidad de recuperar la información hasta unas horas antes de la contingencia.
Replicación de la información	Toda la información generada por un sistema debe ser replicada inmediatamente en sistemas espejos para contar con la posibilidad de otorgar el servicio en una ubicación alterna, igualmente esto daría la posibilidad de recuperar la información hasta el mismo momento de la contingencia.
Procedimientos para la recuperación del sistema en caso de desastre	En el caso de perder el sistema, debe contarse con planes de acción para reinstalar el sistema desde cero, estos procedimientos deben estar lo más simplificados y automatizados posibles para recuperar el sistema en un periodo corto de tiempo.

Tabla 4.2 Acciones preventivas

Todo lo anterior son acciones que una institución debe tomar como practicas diarias, con lo expuesto anteriormente es posible contar con un sistema protegido y garantizar la recuperación en caso de una perdida, claro que implementar todos estos controles es caro, sin embargo siempre es más caro comenzar de cero.

4.3 El WebServer y la zona desmilitarizada

Todos los sistemas distribuidos que tienen salida a Internet están expuestos a ataques que pueden violar su seguridad, pero también están expuestos a todo el mundo, cualquiera con un Web Browser puede en un momento dado ubicarse en el WebServer de una empresa y desde allí intentar ataques al sistema. ¿Qué puede hacer una empresa para protegerse de esto?, afortunadamente mucho, el primer control es un control físico, este control es el conocido como "Zona Desmilitarizada" que es una barrera física que separa la parte interna de una empresa del resto del mundo, es la zona en donde los intrusos pueden actuar pero que por las características de esta zona poco podrían hacer para dañar al sistema.

Estas zonas son un filtro que restringe el paso hacia adentro, generalmente se les ubica entre 2 firewalls (barreras de fuego), la primera barrera lo que hacer es cambiar la dirección IP (Internet Protocol) para que el intruso no vea que segmento de red es el que lo esta atendiendo y de ese punto dirigirse hacia adentro de la empresa; la segunda barrera lo que hacer es cerrar todos los puertos (sockets) y dejar solo abiertos aquellos puertos que atenderán peticiones internamente, esto da una amplia protección ya que los intrusos tendrían que adivinar tanto el segmento de red como el puerto a usar para poder entrar a la red interna de la empresa.

Por otra parte, los elementos que se encuentran en la zona desmilitarizada son Web Servers, y de acuerdo a los planteamientos arquitectónicos expuestos en este trabajo dichos elementos se encargan de realizar la presentación de la información así como de resolver el direccionamiento hacia los elementos internos del sistema, con esto el riesgo de un ataque se reduce el mínimo ya que en el caso de un ataque exitoso, las perdidas serían muy reducidas al tratarse de información estática que es fácilmente recuperable.

4.4 El servidor de aplicaciones

El Servidor de aplicaciones es la pieza medular de los sistemas distribuidos, es el corazón de ellos y es en donde se concentra la mayor parte de la lógica del negocio, en algunos casos comparten funcionalidad y responsabilidades con los Monitores de Transacciones (como Tuxedo por ejemplo) pero su actividad va más allá de la lógica transaccional ya que un servidor de aplicaciones no solo puede realizar las operaciones de una transacción, si no que además es el contenedor de los objetos que tienen la lógica del negocio, es también el encargado de la administración de estos, tiene la responsabilidad de crear las conexiones a las bases de datos, de proporcionar un entorno operativo para los componentes y las aplicaciones, de realizar la replicación de sesiones y ser capaz de tolerar fallas en si mismo o en elementos externos con los cuales tiene conectividad.

En este punto, la recomendación es usar un servidor aplicativo que cumpla con el estándar Java 2 estándar Edition, mejor conocido como J2EE, al momento de escribir este trabajo, el servidor de aplicaciones que mejor implementa este estándar es WebLogic Server de BEA Systems, y de hecho es el líder en el mercado, existen otras opciones como lo es Websphere de IBM, que aunque son productos caros bien valen lo que cuestan, con ellos es posible tener balanceo, tolerancia a fallas, conexiones a bases de datos, sistemas de seguridad que utilicen SSL o listas de control de acceso, monitoreo usando Simple Network Management Protocol (SNMP) consola de administración, control de usuarios, clusters de servidores, replicación de sesiones, manejo automático de threads, cache de sentencias Estándar Query Lenguaje (SQL) hacia la bases de datos, rotación de bitácoras y hasta reinicio automático de los servidores en caso de contingencia entre muchas otras opciones.

En capítulos anteriores se ha expuesto porque y para que se necesita un Servidor de Aplicaciones, y este es elemento sobre el cual se ha hablado a lo largo de este trabajo por lo cual lo dejaremos hasta aquí, basta recordar que el servidor de aplicaciones es el ambiente sobre el cual se ejecutarán nuestras aplicaciones distribuidas y es el corazón del todo el sistema distribuido.

4.5 El acceso a los datos

Dentro del Servidor aplicativo se tiene la posibilidad de crear conexiones a la base de datos y crear elementos que realicen modificaciones a la base de datos, los primeros son conocidos como pool de conexiones y los segundos como Enterprise Java Beans de tipo Entity (Entity EJB's), se tienen varios elementos que actúan sobre la base de datos, pero los básicos y más recomendables son los descritos en la tabla 4.3.

Elemento	Funcionalidad
Pool de conexiones ODBC	Permiten la conectividad hacia la base de datos, se basan en la reutilización de recursos por lo que N usuarios comparten un número reducido de conexiones, esto evita la saturación del ancho de banda además de que mejora el desempeño al evitar la creación de la conexión y la liberación de esta por cada usuario que necesita información de la base de datos.
Multipool de conexiones ODBC	El multipool de conexiones es una colección de pools, la ventaja con este tipo de recurso es que se pueden tener más de una base de datos y direccionarlas desde un solo recurso.
DSN	El Data Source Name da la posibilidad de dar un nombre a una serie de dígitos, haciendo más fácil y manejable el uso de los pools y multipools.
Entity EJB's	Los Entity EJB's son objetos que fueron diseñados para actuar sobre las bases de datos, pero aun cuando tienen ciertas limitaciones son bastante recomendables para hacer altas, bajas y cambios a las tablas de una base de datos.

Tabla 4.3 Elementos para base de datos

En este punto se debe aclarar que se han mencionado dos diferentes tipos de componentes, los que acceden a los datos y los que manipulan los datos, sin embargo es muy importante mencionar que existe otro tipo de elementos que son los elementos que también manipulan los datos pero de una forma transaccional, es decir, implementando el Two Face Commit, en los servidores de aplicaciones aún no se cubre al 100% este tipo de funcionalidad, sin embargo es posible emularla, así por ejemplo un EJB de Sesión puede manipular la respuesta ofrecida por EJB's tipo Entity para realizar transacciones y en su caso hacer el Roll Back si se encuentra alguna inconsistencia o una de las operaciones de la transacción no pudo llevarse a cabo.

El no poder soportar el Two Face Commit al 100% es una limitante que ha llevado a las empresas a utilizar los monitores de transacciones para realizar toda la función de transaccionalidad que se necesita en un sistema distribuido, el mejor y más exitoso ejemplo de este tipo de elementos es Tuxedo, que lleva más de quince años en el mercado realizando las transacciones de múltiples empresas a nivel mundial, un ejemplo de ello es VISA, la cual con ayuda de este monitor de transacciones realiza diez mil transacciones por segundo en promedio.

4.6 Los datos

Esta es finalmente la última capa, es en esta capa en donde reside uno de los activos más importantes de la empresa: la información, y es en esta capa en donde la mayoría de las empresas realizan la menor inversión, por ello en este trabajo se recomienda que sobre esta capa se realicen las acciones de la tabla 4.4.

Acción	Justificación
Respaldo periódico de la base de datos	Se asegura contar con un respaldo duro de la información, la desventaja es que no será un respaldo al 100%.
Replicación de los datos	Se asegura que las bases de datos contendrán la misma información en todo momento, la desventaja es que es un método que incrementa considerablemente la actividad en la base de datos, sin embargo el beneficio generalmente es mayor.
En lo posible, evitar la distribución de la información.	En sistemas distribuidos siempre será mejor tener pocas bases de datos tipo "espejo" replicadas y respaldadas, en donde cada una de ellas pueda tomar las responsabilidades de su espejo en caso de contingencia. Esto es mejor que tener pedazos de bases de datos distribuidos geográficamente, en donde la pérdida de una base de datos lleva a la interrupción del servicio en una zona geográfica determinada.
Mantener el menor número de conexiones, posibles	Esto mejora el rendimiento de la base de datos al no tener que estar creando y destruyendo las conexiones de los usuarios hacia la base de datos, es mejor actuar con conexiones establecidas que no cambien y dedicar el poder de procesamiento a la búsqueda y modificación de datos.
Tener procedimientos almacenados	Un procedimiento almacenado al estar compilado en la base de datos tiene un mejor desempeño que un query que es ejecutado por primera vez. Si esto sé transada a miles de operaciones, el beneficio es muy grande.

Tabla 4.4 Acciones recomendadas sobre los datos

Estas son pequeñas recomendaciones que pueden ayudar mucho en la administración de una base de datos, sin embargo no se ha mencionado nada sobre la normalización de esta, y es que en este trabajo se asume que la base de datos ya está normalizada.

CAPÍTULO 5

LAS PRUEBAS, LA OPERACIÓN, EL MONITOREO Y CONTRATOS PARA EL SISTEMA DISTRIBUIDO

Una vez que el sistema se ha creado y construido es necesario hacerlo pasar por el esquema de Aseguramiento de la Calidad, realizar las pruebas necesarias, y asegurarse de contar con los acuerdos necesarios por escrito para poner al nuevo sistema a trabajar, sin embargo el punto importante no es solo ponerlo a operar y que entregue los resultados esperados sino por el contrario que el nivel de servicio sea el esperado por los usuarios.

5.1 El aseguramiento de la calidad

Probablemente este tema suene muy tratado ya que todos los sistemas tradicionales (cliente / servidor por ejemplo) deben realizar pruebas unitarias para verificar que los resultados entregados son los correctos, además se deben realizar pruebas piloto para verificar que el sistema se comporta como es debido en un ambiente de operación real, sin embargo para asegurar la calidad de un sistema distribuido es necesario ir más allá de las pruebas mencionadas anteriormente, y es que dada la división de capas en las que esta construido un sistema de este tipo, la conectividad necesaria y la reusabilidad de elementos y componentes se debe asegurar la correcta convivencia con el resto de los sistemas que estén actualmente en producción.

Esta no es una tarea fácil, es necesario convocar a pruebas a todos los responsables de los sistemas, es necesario crear scripts de pruebas para cada sistema y probarlos con todos los sistemas operando, puede sonar repetitivo y hasta una perdida de tiempo el probar nuevamente un sistema si ya esta operando correctamente en producción, sin embargo esto no es un punto opcional, debe recordarse que los sistemas actúan sobre plataformas operativas que están distribuidas y son compartidas, es en este punto en donde se encuentra otro punto fundamental de los sistemas distribuidos, los sistemas no deben considerarse ya como sistemas aislados e independientes, si no como Sistemas de Servicios dependientes unos de otros, esto se debe a que las plataformas operativas en donde corren (los servidores aplicativos) se basan en la reutilización de servicios y componentes.

El cambio de paradigma es fundamental y hasta no lograr asimilarlo no será posible obtener éxito en la puesta en producción de un sistema de este tipo, por esto las pruebas enlistadas en la tabla 5.1 deben realizarse de la manera en que se ha mencionado, de lo contrario se corre el riesgo de tener múltiples fallas en producción y es que en un ambiente operativo en donde se comparten componentes de una aplicación siempre se pueden esperar fallas en donde menos se ha pensado.

Prueba	Resultado
Scripts de pruebas	Lista de pruebas y resultados esperados, se espera comprobar la correcta entrega de resultados
Pruebas unitarias	Comprobar entrega correcta de resultados operativos
Pruebas de convivencia	Comportamiento adecuado del nuevo sistema distribuido así como de los sistemas actualmente en producción
Pruebas de estrés	Comportamiento adecuado del nuevo sistema distribuido así como de los sistemas actualmente en producción bajo un entorno operativo en condiciones de horas y días pico
Prueba piloto	Liberación paulatina del sistema a producción, los resultados deberán comprobarse ya en el ambiente de producción.
Liberación a producción	La puesta en todos los ambientes de producción del nuevo sistema

Tabla 5.1 Las pruebas

Las pruebas de estrés deben realizarse en el ambiente de producción con ayuda de software especializado y diseñado para este propósito, estas pruebas ayudan a comprobar si los recursos de hardware, de software así como el propio diseño y arquitectura de la aplicación serán adecuados para la demanda esperada. Un sistema puede entregar los resultados esperados y comportarse de la manera correcta, pero si el sistema niega el servicio debido a carga excesiva entonces de nada habrá servido contar con el mejor de los diseños.

La negación del servicio por carga excesiva suele ocurrir en los horarios y días pico, pero deben tomarse en cuenta también que son factores a considerar los procesos operativos como respaldos, procesos batch o transferencia de archivos, ya que estos últimos al no programarse en los horarios adecuados pueden causar problemas en la operación al saturar los recursos.

Aunque es de vital importancia realizar la prueba piloto no siempre sucede así, y esto se debe a la gran presión que existe por sacar el nuevo producto, sin embargo el no realizar una prueba piloto puede generar errores que conllevan a la perdida del prestigio o hasta a la perdida de clientes. Las pruebas piloto son también muy costosas y hasta lentas ya que debe tenerse personal realizando una parte de las operaciones reales en el nuevo sistema, verificar resultados y compararlos contra lo esperado o contra el sistema a sustituir, pero esto no es todo, deben verificarse también los resultados del resto de los sistemas que ya están en producción y que de alguna manera pueden verse afectados al estar relacionados con el nuevo sistema a liberar.

Cuando todo lo anterior se cubre de manera satisfactoria es entonces cuando la gerencia puede dar el visto bueno para finalmente liberar a producción el nuevo sistema, sabiendo que al distribuirlo masivamente a todo el ambiente operativo no tendrá problemas, o al menos la probabilidad de tenerlos será mínima.

5.2 La planeación de capacidades

Al momento de concebir un nuevo sistema distribuido, es necesario considerar si la carga esperada podrá ser cubierta con los recursos de hardware, software y de comunicaciones con los que se cuenta actualmente. Para ello se deben realizar estimaciones por el área de negocio en donde se debe llegar a conocer el número de usuarios concurrentes, las horas pico, los días pico y las transacciones por segundo esperadas, las máximas y el promedio, en la tabla 5.2 se muestran algunas de las métricas más importantes a considerar.

Métrica	Descripción
Usuarios concurrentes	El número máximo de usuarios concurrentes en un mismo tiempo
Horas pico	El horario en el que se espera contar con el número máximo de usuarios concurrentes
Días pico	Los días con mayor afluencia de usuarios
Tiempo de transacción	El tiempo máximo esperado para una transacción
Transacciones por segundo	El número máximo de transacciones en un mismo segundo
Bytes por segundo	Los bytes máximos a transferir
Espacio en disco	El espacio en disco mínimo requerido para la operación de l sistema
Memoria disponible	La memoria total que se espera utilice el sistema
Uso de procesador	El porcentaje de procesador que se espera utilice el sistema

Tabla 5.2 Métricas para planeación de capacidades

Todas las métricas anteriores son importantes al momento de la planeación ya que al compararlas y prorratearlas contra lo que se tiene actualmente darán la señal de si es o no necesario adquirir más poder de procesamiento, si será necesario invertir en equipo de comunicaciones para que el ancho de banda sea suficiente o si las capacidades de procesador, disco o memoria deben ser ampliadas.

Lo anterior lleva a una sola cosa que se ha mencionado como una caracteristica de los sistemas distribuidos: la escalabilidad, de lo que se ha hablado es de la posibilidad de agregar una caja más al entorno operativo, conectarla y poner a trabajar al sistema, y es precisamente de eso de lo que se esta hablando, si la planeación de capacidades da como resultado que los recursos actuales son insuficientes para soportar la demanda, entonces solo habrá que comprar más recursos de las mismas características y agregarlos al arreglo de procesamiento con el que ya se cuenta.

5.3 Los contratos de servicio

Al momento de salir a producción con un sistema distribuido debemos recordar el cambio de paradigma de Sistema a Sistema de Servicios y no perder de vista que un nuevo servicio que se ofrece al cliente es una serie encadenada de pequeños servicios distribuidos en capas, con esto en mente se debe proceder a obtener los contratos de servicio necesarios con todos los responsables de la operación del sistema, esto quiere

decir que será necesario establecer convenios con las áreas de operación, soporte a la producción, monitoreo, respaldos y hasta con él área de mantenimiento operativo.

Los contratos de servicio son elementos que indican cual es el nivel de servicio esperado por el área de negocio, en ellos se deben especificar los tiempos de operación (que generalmente son de 7 x 24), se detallas los procesos de operación, las acciones a seguir en caso de contingencia, las capacidades que debe tener un especialista que soporte a la producción, el escalamiento en caso de problemas, los elementos a monitorear especificando sus umbrales y acciones automáticas.

En resumen, los contratos de servicio son documentos en donde se especifica el nivel de servicio y respuesta que deben entregar las áreas operativas a las áreas de negocio con la finalidad de darles a los clientes alta disponibilidad.

5.4 La operación del sistema

Al momento de entregar un sistema sea este distribuido o no, se deben entregar los manuales técnicos y de usuario, pero además para el exitoso funcionamiento de este debe estregarse un manual de operación, en este manual se detallan todos los procesos operativos que deben realizarse, describiendo cada uno de los procesos a ejecutar con horarios y frecuencias.

El o los manuales de operación son documentos dirigidos a personal que no es experto en el sistema, muchas veces ni siquiera sabe cual es la función del sistema y muchas otras ni siquiera tiene que ejecutar el 100% de las comandos de operación para todo el sistema, si no por el contrario para una parte de este únicamente.

Debe recordarse que los operadores cubren una función de guardias y que su labor esta dividida en turnos que cubren las 24 horas los 7 días de la semana, los 365 días del año, son quienes se encargan de vigilar el sistema y hacerlo funcionar, de atenderlo y ejecutar los comandos básicos para que el sistema este operable, por estas razones un manual de operación debe decir exactamente lo que se debe hacer, especificando las maquinas, las direcciones IP, los puertos, las rutas, los horarios, las frecuencias y los comandos especificos a ejecutar, es decir, un manual de operación debe estar digerido y facilitar lo más posible la labor del operador.

Al liberar un sistema distribuido, los manuales de operación se convierten en una herramienta indispensable para el funcionamiento de estos, además deben de estar segmentados por plataforma para que cada capa funcione efectivamente y pueda entregarle a la siguiente capa la información y se mantenga en optimas condiciones.

5.5 El monitoreo

El monitoreo de un sistema es otra parte fundamental de los sistemas distribuidos, el objetivo del monitoreo es evitar los problemas, y no como mucho se ha pensado el darse

cuenta de ellos. En la mayoría de los casos en donde no existe el monitoreo o este es deficiente, quien detecta el problema es el usuario final, y al suceder esto la imagen del negocio decae ya que al darse un error generalmente se estará negando el servicio.

El monitoreo preventivo debe vigilar los signos vitales del sistema, comenzando desde el espacio en disco necesario y terminando en el más escondido de los signos o elementos imprescindibles del sistema.

Los signos vitales de una máquina son los siguientes:

1.- Espacio en disco
2.- Uso del procesador
3.- Uso de la memoria
4.- Threads disponibles
5.- Nivel de swap

Los signos vitales de un sistema distribuido son:

1.- La página principal
2.- La conectividad a la base de datos
3.- La conectividad entre capas
4.- El uso de memoria asignada
5.- Los threads disponibles

Existe un tercer nivel de signos vitales, dicho nivel se refiere a la disponibilidad del medio, es decir de las comunicaciones, de los balanceadores, de los firewalls, de los pix y de los ruteadores, de la red en general.

Lo importante de un contrato para el monitoreo es que se deben monitorear todos los signos y elementos vitales para la correcta operación del sistema, pero no en un enfoque correctivo, si no en un enfoque preventivo con el afán de evitar los problemas, y no tener como fin el solucionarlos.

5.6 Los sistemas de respaldo

Todo contrato de servicio debe tener especificado cuáles son los elementos a respaldar en cada una de las capas, de igual manera debe especificar que elementos o bitácoras deben vaciarse o borrarse y especificar en que momento se debe hacer esto, por ejemplo, en un sistema distribuido, para cada capa deben respaldarse los siguientes elementos:

1.- El software original.
2.- La ruta en donde reside la configuración dinámica del software.
3.- Las bitácoras de acceso.
4.- Las bitácoras de errores.
5.- Las bitácoras generales del sistema.
6.- Las bitácoras de auditoria.
7.- Las bitácoras de transacciones.

De la misma manera debe especificarse la periodicidad de estos respaldos (por hora, diario, semanal, mensual, etc.) y la localidad que los resguardará. Por otra parte también es necesario indicar que sucederá con los respaldos, si estos deben ser verificados antes de ser resguardados y cual será la rotación de estos y la reutilización de los medios magnéticos en donde se realicen.

Un buen contrato de servicio especifica en la parte de respaldos que todo el sistema y sus evidencias deben ser respaldados, pero un contrato excelente es aquel que indica los tiempos en los que el área operativa debe entregar un respaldo.

5.7 La contingencia operativa

Este es el punto final de los sistemas distribuidos, la contingencia operativa, desde el principio de este trabajo se ha hablado de ella como una característica de este tipo de sistemas y aquí es donde se mide la efectividad de todo el trabajo de diseño y arquitectónico.

Cuando un sistema entra en contingencia y este sistema esta bien planeado, el usuario final no notará que internamente el negocio tiene fallas técnicas, el usuario no detectará que quien le esta contestando no es el arreglo de servidores que atiende a su cobertura, probablemente notará una pequeña dilatación en el tiempo de atención pero nada que afecte su uso normal.

Un sistema distribuido bien planeado, bien construido y bien implementado contará casi por característica propia con una contingencia operativa que le permitirá trabajar aún con la mitad de sus recursos de cómputo. Se dice que ni siquiera desastres naturales como terremotos o inundaciones deberían afectar a un sistema distribuido bien planeado, ¿por qué?, muy sencillo, un sistema distribuido además de separar en capas su funcionamiento también tiene contingencia local, pero además por sus mismas características puede y debe contar con una contingencia regional, es decir, también debe distribuir su funcionamiento a otra entidad que no pueda verse afectada por fenómenos naturales que ocurran localmente en la localidad principal.

Todo esto en conjunto debe ponerse por escrito en un convenio de servicio y constituye el objetivo de todo sistema distribuido: estar arriba y corriendo 7 x 24.

CONCLUSIONES

En este trabajo se ha hablado de los sistemas distribuidos, de cómo se ha evolucionado desde los sistemas monolíticos, pasando por los sistemas Cliente-Servidor, luego los sistemas de publicación y finalmente lo que se ha convertido en las aplicaciones distribuidas.

Se ha planteado el porqué este tipo de aplicaciones son complicadas de construir y mucho más complejas de operar que las aplicaciones cliente servidor o aplicaciones legacy, pero lo que se pretende resaltar es el hecho de que un sistema distribuido es una cadena de valor en el que intervienen múltiples factores, así como múltiples elementos que hacen que el sistema tenga éxito o sea un rotundo fracaso. La responsabilidad de este tipo de sistemas ya no depende solamente de los analistas, diseñadores y desarrolladores, ahora es necesario llevar más allá la responsabilidad del sistema ya que un sistema distribuido necesita una postimplementación, un seguimiento, un mantenimiento, pero no solo de código, sino también operativo, es necesario obtener métricas de servicio, realizar un monitoreo constante, medir los tiempos de respuesta, realizar el cálculo de capacidades antes y después de la liberación para que de ser necesario el sistema sea escalado.

La palabra escalabilidad es uno de los principales objetivos de un sistema distribuido, y se puede decir que si un sistema es fácil de escalar entonces todo el trabajo del que se ha hablado en este documento ha sido llevado a buen término, y es que debe recordarse el porqué los sistemas Cliente Servidor han dejado de ser la respuesta a las crecientes demandas del mercado, por ejemplo su disponibilidad. Este factor ha llevado a las corporaciones a replantearse sus objetivos en términos de tecnología de la información y pensar en construir sistemas distribuidos que representen servicios de negocio, ya que hoy día no es suficiente contar con resultados correctos en términos de cálculo, ni es suficiente dar servicio en horarios hábiles, hoy es requisito dar un servicio 7 días por 24 horas, los 365 días del año, y quizás el reto mayor es el de dar el servicio a todos los clientes, a cualquier hora, cualquier día y sin importar el lugar de residencia del cliente.

También se observo que los sistemas distribuidos ayudan a las corporaciones a actualizar la versión de estas aplicaciones más fácilmente y sin necesidad de interrumpir el servicio ya que un sistema distribuido con una buena actualización de versiones puede dar servicio ininterrumpido mientras es actualizado. Como hemos visto esto puede lograrse al apagar y luego actualizar una parte de los componentes mientras la otra parte continua trabajando y una vez realizada la actualización en la partición primaria se puede continuar con la actualización secundaria mientras la nueva versión implementada en la primera partición da el servicio ya en su nueva versión. La fórmula como se ha descrito consiste en crear una arquitectura distribuida en múltiples capas que distribuya la funcionalidad y las responsabilidades del sistema entre diferentes objetos y capas, con esto es posible armar múltiples sistemas aprovechando la reusabilidad de componentes al distribuir la funcionalidad en todos los sistemas, contribuyendo de esta manera a la reducción de costos y más importante aún, a la reducción del tiempo en el que un sistema es modificado y se libera a producción para adaptarse a las nuevas necesidades del mercado.

En este trabajo se observo que los sistemas distribuidos basados en capas y componentes además de tener una gran flexibilidad para realizar actualizaciones y crear nuevos sistemas, proveen la posibilidad de crear conexiones con aliados de negocios, esto es un beneficio tanto del diseño como de la tecnología en la que se basan los sistemas distribuidos. Con esto es posible ofrecer un servicio a un socio de negocio, permitiéndole contar con algunos servicios que le ayuden a ofrecer sus propios servicios a sus clientes, acrecentando de esta manera la cadena de valor y creando nuevos negocios al realizar transacciones automáticas con sistemas de socios de negocio (terceros). Es claro que puede decirse que esto es posible hacerlo aún con aplicaciones legacy, o con aplicaciones cliente servidor, y nada más alejado de la verdad, ¿pero cuántas modificaciones?, ¿cuántas lineas de código?, ¿cuántos rediseños?, ¿cuánto gasto tendria que hacerse para poder conectarse con un socio de negocio?, la respuesta es simple: mucho. Por esto, los sistemas distribuidos ofrecen flexibilidad, pues solo es necesario tomar el objeto u objetos que ofrecen el servicio, ajustarle la seguridad, maquillar el servicio y listo, nuestro socio puede utilizar nuestro servicio.

La seguridad es otro factor muy importante, ahora ya el esquema de usuario y password no es suficiente, no solo basta autenticar al usuario, es necesario proteger su información, tanto de agentes externos como de agentes internos. Por ello los sistemas distribuidos se construyen sobre tecnologia que provee este tipo de mecanismos de seguridad, los servidores aplicativos tienen la capacidad de proteger la información utilizando algoritmos de encripción, pero también tienen la capacidad de otorgar niveles de acceso a los usuarios y aplicaciones a nivel de componente. Con este nivel de seguridad se vuelve realmente difícil que la seguridad sea violada, pero desafortunadamente aún no se crea el sistema con la perfecta seguridad.

En este trabajo se ha abogado por los sistemas distribuidos y se ha hecho énfasis en tratar de resaltar los retos que impone crear, diseñar, mantener y operar un sistema de estas características. Por otro lado también se han expuesto las ventajas competitivas que ofrecen este tipo de sistemas, sin embargo el mayor conocimiento que el lector puede obtener de este trabajo más allá de lo que se ha detallado, es el cambiarle el paradigma, y el objetivo se habrá cumplido si se ha logrado transmitir el concepto de que un sistema distribuido ya no debe visualizarse ni pensarse como un sistema únicamente, si no como un servicio, ya que finalmente eso es lo que se le ofrece y requiere el usuario de cualquier tipo de negocio: servicios.

Es por esto que el planteamiento de que la visualización, diseño, construcción, implantación, post implantación, mantenimiento, monitoreo y seguimiento de un sistema ya no debe realizarse teniendo en mente solo sistemas de información, sinó también servicios de información, es decir, servicios que satisfagan oportunamente las necesidades de la razón de ser de los negocios: los clientes.

Por último se plantea la posibilidad de aprovechar este trabajo para analizar la viabilidad de crear una herramienta que ayude a los desarrolladores de sistemas distribuidos a crear más fácilmente este tipo de sistemas al incluirle las nuevas actualizaciones que se realicen a los estándares J2EE.

BIBLIOGRAFÍA

Joe Zuffoletto, Gary Wells, Brian Gill, Geoff Schneider, Barrett Tucker, Rich Helton, Michael Madrid, Sunil Makhijani. (2004). ***BEA Weblogic(R) Server Bible,*** (Second Edition). Indianapolis, IN 46256. Wiley Publishing, Inc.

Deepak Alur, John Crupi, Dan Malks. (2003) ***Core J2EE Patterns: Best Practices and Design Strategies***, (Second Edition). Upper Saddle River, NJ 07458. Prentice Hall PTR & SUN Microsystems Press.

Michael Girdley, Rob Woollen, Sandra L. Emerson. (2001) ***J2EE Applications and BEA WebLogic Server***, (First Edition). Upper Saddle River, NJ 07458. Prentice Hall PTR.

Budi Kurniawan. (2002). ***Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions***, (First Edition). New Riders Publishing.

Greg Nyberg (2003). ***WebLogic Server 6.1 Workbook for Enterprise JavaBeans,*** (3rd Edition). Titan Books.

Mark Cade, Simon Roberts. (2002). ***Sun Certified Enterprise Architect for J2EE Technology Study Guide***, (First Edition). San Antonio Road, Palo alto California 94303. SUN Microsystems Inc.

Peter Zadrozny, Francisco Gomez. (2000). ***Professional J2EE Programming with BEA WebLogic Server,*** (First Edition). Wrox Press.

Sean Christofferson, Srinivas Jayanthi, Steven Traut, Wira Pradjinata. (2003). ***BEA WebLogic Workshop: Building the Next Generation Web Services Visually, (First Edition).*** Indianapolis, IN 46256. Wiley Publishing, Inc.

Khawar Zaman Ahmed, Cary E. Umrysh. (2001). ***Developing Enterprise Java Applications with J2EE and UML***, (Fisrt Edition). Booch, Jacobson, Rumbaugh Series Editors. Addison Wesley Object Technology Series.

Kaminaris Stephanie Fesler, Albert Saganich, Al Saganich, Jr., Albert J. Saganich. (2001). ***Developing Enterprise Applications with BEA WebLogic Server***, (First Edition). SAMS Publishing.

Inderjeet Singh, Beth Stearns, Mark Johnson, Enterprise Team. ***Designing Enterprise Applications with the J2EE(TM) Platform (2nd Edition).*** New York. Addison-Wesley.

Stefan Sigfried. (1995). ***Understanding Object Oriented Software Engineering,*** (First Edition). Los Alamitos, CA, IEEE Computer Society Press.

Dave Collins. (1995). ***Designing Object Oriented User Interfaces***, (First Edition). Redwood City, CA 94065, Benjamín Commings Publishing Company, Inc.