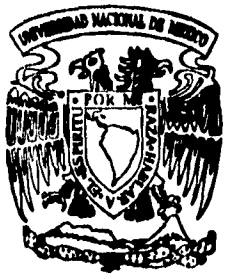
22
2EJ



UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

CLIENTE / SERVIDOR, UN AUTENTICO CAMBIO EN TECNOLOGIA INFORMATICA Y SU APLICACION EN UN CAJERO BANCARIO.

SEMINARIO DE INVESTIGACION INFORMATICA

Que para obtener el Título de:
LICENCIADO EN INFORMATICA

P r e s e n t a n:

BLANCA GRACIELA LOPEZ JIMENEZ
MARCO ANTONIO CAMACHO MORENO
RENE SAUL DE LA ROSA CASTILLA

Asesor del Seminario:
L.C. y M. en C. Marina Toriz Garcia

México, D.F. 1995

FALLA DE ORIGEN

Al finalizar este trabajo queremos dejar constancia y agradecimiento de la labor y el gran apoyo que recibimos por parte de la L. C. y M. en C. Marina Toríz García quien fungió como asesor de este seminario de investigación.

El haber concluido este periodo de estudios con el presente Seminario de Investigación representa todos los esfuerzos de varias personas a las que deseo dedicarles el presente documento.

Principalmente a mis padres con todo mi cariño y admiración: Profra. Graciela Jiménez Zaldívar y Prof. Heliodoro L. López Angeles por su ejemplo de lucha y entrega incondicional y por que gracias a su esfuerzo he salido adelante.

A mi esposo Lic. Francisco Javier Villegas Landín con todo mi amor, ya que ha representado un gran apoyo (profesional y personal) y por todo el amor que me ha brindado.

A la pequeña que me ha dado una nueva perspectiva de la vida y a quien dedicaré todo mi esfuerzo para darle un mundo mejor, a mi hija con todo mi amor Arantxa Berenice.

A mis hermanos: L. Adriana, Heliodoro e Israel que siempre estuvieron en el momento preciso.

A mis queridos sobrinos : Carlos Alberto y Arón Israel.

A Dios que me dió la oportunidad de llegar a este momento con los seres que más aprecio y a la Virgen de Guadalupe que me acompañan siempre con su infinito amor.

A mis amigos y compañeros que han demostrado la grandeza de la amistad Marco y René.

GRACIAS por ser quienes son.

Blanca Graciela.

A quienes me dieron su apoyo, su estímulo y demostraron siempre su inquebrantable confianza en mí, sin la cual no habría llegado a la cima; a mis padres: Sr. Aureliano Camacho Carrillo y Sra. Audelia Moreno de Camacho.

A quienes comparten conmigo la grandeza de ser astillas del mismo árbol; a mis hermanos: Primitivo, Marisela y Rosalinda.

A la Universidad Nacional Autónoma de México por brindarme la oportunidad de cultivar mi espíritu.

A Blanca, Ana Luisa, Marisela, Claudia y René; por las horas de trabajo, los momentos de diversión, los instantes de tristeza y la infinita amistad que nos une.

A quien ocupa un lugar especial en mi corazón: Verónica.

Al Lic. Francisco Javier Villegas Landín, por su valiosa cooperación para la realización del presente seminario.

A mis abuelos y a la Sra. Anatolia Moreno de Camacho, quienes a pesar de su partida existirán mientras viva.

A ustedes...

...Gracias.

Marco Antonio.

Este trabajo representa la culminación de una etapa muy importante, la cual encierra el esfuerzo, dedicación y apoyo de muchas personas a las cuales solo puedo retribuirles con mi más sincero agradecimiento.

A mi mamá Margarita por ser la persona que más ha dejado y brindado por mi, y de la cual estaré eternamente agradecido.

A mi tío Saúl en el cual encontre siempre el apoyo de un verdadero padre.

A toda mi familia que siempre ha estado conmigo y que de una u otra forma me ha apoyado.

A Mónica por su amor, comprensión y por todo este tiempo a mi lado.

A todos mis amigos que han compartido los buenos momentos y acompañado en los dificiles.

Al Lic. Francisco Villegas por su ayuda para la realización de este trabajo.

A Dios por brindarme la oportunidad de llegar a este momento y por todo lo que he recibido en la vida.

René.

INDICE

INDICE DE FIGURAS

Antecedentes

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

INDICE DE GRAFICAS

Capítulo 1

Capítulo 5

INDICE DE TABLAS

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

INTRODUCCION

Los años 90' s han demostrado en la industria informática un cambio radical en cuanto a la forma de trabajo. Esto se debe en parte a las facilidades que a nivel de comunicaciones existen en México y otras partes del mundo, aunadas a la gran cantidad de recursos informáticos.

Esto ha permitido que diversos grupos de trabajo interactúen aún cuando éstos estén distribuidos en lugares geográficamente distintos.

Tales cambios han provocado el surgimiento de nuevos conceptos en el mundo de la Informática. Uno de estos conceptos es el de Cliente/Servidor, el cual ha generado diversas expectativas e incertidumbres sobre los beneficios, desventajas y lo que implica llevar a cabo la implementación de una Arquitectura Cliente/Servidor. De esta forma podemos encontrar a diario diversos puntos de vista sobre este tema en publicaciones, conferencias y empresas que dedican recursos y esfuerzos para presentar y vender este concepto.

El objetivo del presente Seminario de Investigación "Cliente/Servidor, un Auténtico Cambio en Tecnología Informática y su Aplicación en un Cajero Bancario" es enriquecer al lector en el conocimiento de la Arquitectura Cliente/Servidor, ¿qué es?, ¿cuáles son sus elementos funcionales?, ¿cómo operan?, ¿cuál ha sido su impacto en el procesamiento de datos? e implementar una Arquitectura de este tipo a través de la construcción de una aplicación en una de las áreas funcionales de una sucursal bancaria.

Para explicar en forma sencilla el concepto Cliente/Servidor podemos hacer una equivalencia en la relación que existe entre el cliente de un restaurante, un mesero, y el cheff en la cocina. Al pedir un platillo, el cliente le pide al mesero (enlace de comunicación entre el cliente y el servidor o cheff) los platillos que desea. Después de haber hecho su petición, el cliente se despreocupa de la solicitud que hizo, pudiéndose avocar a otras tareas, como conversar con otros miembros de la mesa, leer el periódico, etc. . Por su parte, el mesero, que es el medio de comunicación entre el cliente y el cheff o servidor, lleva las peticiones del cliente, a manera de instrucciones a este último, el cual las procesa o las lleva a cabo. Al término del procesamiento, cuando los platillos están listos, el servidor

hace acceso del medio de comunicación o mesero y envía la comida o resultado al cliente. El mesero sabe que pidió cada cliente, además sabe en que orden deben llevarse los platillos, mismos que sirve al cliente. Para ese entonces, el cliente recibe los platillos, y a su vez los procesa por su cuenta, ingiriéndolos.

En este esquema vemos que el cliente no está en continua interacción con el mesero, a su vez, no es necesario recordar al mesero en todo momento que pasa y mucho menos debe ir a la cocina y hablar con el cheff. En otras palabras, efectúa actividades diversas y sólo recibe sus órdenes ya procesadas por el servidor. Sin embargo, el llevar a la práctica este concepto involucra el conocer y conjuntar diversos elementos, los cuales tratamos en el presente seminario de investigación que se integra por un apartado de antecedentes y cinco capítulos, de los cuales los primeros cuatro están dedicados a presentar y describir todos los aspectos que involucra la Arquitectura Cliente/Servidor para que en el último se puedan aplicar estos aspectos en la documentación del caso práctico que se mencionó anteriormente. La última parte del seminario está integrada por nuestras conclusiones, los apéndices, el glosario y la bibliografía.

Se presenta un índice de capítulos donde se resalta el título de cada uno con negritas y los subtítulos con mayúsculas. Se incluyen también índices de figuras, tablas y gráficas cuya numeración se integra por el número del capítulo que corresponde y un número secuencial, este último se inicia desde el 1 en cada capítulo, seguidos por el nombre así como del número de página donde se encuentra.

En los antecedentes, explicamos brevemente la evolución de los sistemas de cómputo, desde los centralizados basados en un Mainframe, quien es el responsable de determinar qué consultas pertenecen a qué usuario, de obtener la lógica del negocio y de recuperar los datos. Destacamos las ventajas de este tipo de procesamiento como un gran desempeño, integridad de datos bien protegida y gran confiabilidad; en tanto que como desventajas están que se tiene una interface poco amigable y limitada para muchas aplicaciones y sustancial carga en el mainframe, por ser éste el único recurso de procesamiento inteligente. Esto propicio el desarrollo de las redes de área local (LAN) buscando aprovechar la inteligencia de las estaciones de trabajo, con un Servidor de Archivos que simplemente tiene la

función de permitir compartir archivos y periféricos de las mismas, obteniendo como ventajas, de utilizar el poder de las PC's como estaciones de trabajo e incluso incrementar éste agregando otras PC's a la red, la reducción de costos en comparación con un mainframe es un punto importante. Sin embargo, por la degradación de la velocidad de comunicación entre los equipos por el movimiento de un sentido a otro de la red de grandes volúmenes de información, el no contar con un servidor inteligente que proteja la integridad de los datos, así como la complejidad del control y administración de la red dieron lugar al surgimiento del concepto de Cliente/Servidor que trata tanto al servidor como a las estaciones de trabajo como dispositivos inteligentes y programables explotando todo el poder de cómputo de cada uno de éstos.

En el capítulo 1 "Ambiente Distribuido", describimos el ambiente de cómputo en el cual se ubica una Arquitectura Cliente/Servidor. El hablar de estándares nos facilita la comprensión de la creación de sistemas interoperables y portables siendo la X/Open y la Open Software Foundation (OSF) organizaciones de estándares que han tenido fuerte impacto en Cliente/Servidor. El ambiente de cómputo distribuido (DCE) es un concepto desarrollado por la OSF que abarca un conjunto integrado de sistemas operativos y de servicios independientes de la red que soportan el desarrollo, uso y mantenimiento de las aplicaciones distribuidas. DCE tiene capacidad para ofrecer una red administrable, transparente e interoperable de sistemas de diferentes plataformas y diversos fabricantes. Actualmente DCE consiste de siete herramientas y servicios que están divididos en 1.Servicios Distribuidos Fundamentales integrados por: Multithread, RPC, Servicios de Directorio Distribuido, Servicios de Temporalización y Servicios de Seguridad (Kerberos) y 2.Servicios de Compartición de Datos integrados por: Sistemas de Archivos Distribuidos (DFS) y Soporte para equipos carentes de disco. Todo esto nos lleva a tratar el punto de los Sistemas Distribuidos considerándolos más eficientes que los centralizados por su tolerancia a fallas y por permitir el trabajo en paralelo. Por otra parte la distribución de datos constituye una de las principales ventajas de la Arquitectura Cliente/Servidor al permitir que éstos sean accesados desde estaciones locales o remotas; para fundamentar este punto se presenta la opinión de los autores Bruce Elbert y Bobby Martyna sobre las ventajas de la distribución de datos en términos de costo.

En el capítulo 2 "Componentes de Cliente/Servidor", se analizan los principales componentes de una Arquitectura Cliente/Servidor, los cuales a nivel de funciones son el Front-End y el Back-End. El primero constituye la parte de interpretación y presentación hacia el usuario y reside en la máquina cliente; el segundo está formado por los programas que residen en el servidor y están orientados a atender las solicitudes de cliente.

En un desglose más amplio de los componentes de esta arquitectura, describimos al servidor, el cual se encarga de brindar seguridad, administrar y proveer a los clientes de los recursos de la red y de las bases de datos. Se mencionan también las principales categorías de servidores que existen tales como Servidor de impresión, de comunicaciones, de fax, de correo, de bases de datos, y de archivos, poniendo mayor énfasis en estos últimos y se presenta una gráfica de las principales plataformas empleadas para los servidores, los cuales en base a las necesidades y recursos disponibles pueden ser desde un mainframe hasta una estación de trabajo.

Por otra parte está el cliente, el cual generalmente es una computadora personal, una minicomputadora o una estación de trabajo, que accesa los servicios de la red y la información que reside en los servidores, con la posibilidad de contar con recursos propios. Se mencionan los puntos importantes a considerar en los clientes tales como capacidad de memoria, de disco y la necesidad de contar con interfaces gráficas que permitan hacer amigables las aplicaciones.

En cuanto a los sistemas operativos más comunes para el soporte de la Arquitectura Cliente/Servidor encontramos el sistema de Apple Macintosh, Windows NT de Microsoft, UNIX, Novell NetWare y OS/2 de IBM, cada uno de los cuales guardan diferentes características y potencialidades entre sí.

Como tercer punto fundamental de este capítulo está el protocolo de comunicación empleado, ya que éste es el principal medio de conexión entre las aplicaciones del cliente y los servicios que están disponibles en la red y los servidores; destacan como los protocolos más conocidos TCP/IP, Net Bios y APPC de IBM. Una Arquitectura Cliente/Servidor tiene como característica primordial las facilidades gráficas que se ofrecen al usuario en las máquinas clientes, esto involucra contar con GUIs, las cuales permiten y facilitan la comprensión, manejo y aprendizaje de las aplicaciones. Aunado al concepto de GUIs, está el de APIs y el de Middleware de los cuales el primero facilita el acceso a los servicios disponibles en la red por medio de funciones y llamadas a funciones, mientras que el segundo permite

establecer la interface adecuada con las capas superiores de la red. Estos conceptos están directamente relacionados con las aplicaciones que se manejan en una Arquitectura Cliente/Servidor, por esta razón describimos los componentes de éstas, los cuales son la lógica de presentación, la lógica del negocio, la lógica de datos, y los servicios de comunicaciones.

La última parte del capítulo está dedicada a la seguridad que se requiere manejar en Cliente/Servidor y los principales métodos que existen para la autentificación de usuarios, destacando el sistema kerberos que permite autentificar usuarios y conexiones a una red.

Un elemento básico en la Arquitectura Cliente/Servidor es: la red a emplearse para que se ejecuten las aplicaciones desarrolladas en este ambiente; por esto, en el capítulo 3 "Cliente/Servidor en Dos y Tres partes" analizamos como primer punto, los siguientes tipos de redes: Interconexión de sistemas Abiertos (OSI/ISO), Protocolo de Internet: Transmission Control Protocol/Internet Protocol (TCP/IP) y Arquitectura de redes (IBM/SNA); las cuales por sus características son capaces de satisfacer los requerimientos de un sistema distribuido y en particular de una Arquitectura Cliente/Servidor.

Así como existen diversos tipos de redes, también la Arquitectura Cliente/Servidor posee dos modalidades para su implementación, división basada en el lugar en que se colocan los componentes de una aplicación Cliente/Servidor (Servicios de Presentación, Lógica de Presentación, Lógica del Negocio, Lógica de Datos, Servicios de Datos y Servicios de Archivos). La primera de éstas se denomina "Cliente/Servidor en dos partes", sobre este tema, presentamos sus generalidades y tres alternativas de donde colocar cada componente de la aplicación; además de mencionar los beneficios y desventajas de adoptar cada una de estas opciones y en general de la Arquitectura Cliente/Servidor en dos partes.

Todo elemento informático evoluciona constantemente y Cliente/Servidor sigue esta línea. Buscando eliminar las desventajas de la Arquitectura Cliente/Servidor en dos partes, surge la Arquitectura Cliente/Servidor en tres partes; siendo éste el último punto tratado en el capítulo, del cual se analizan sus generalidades, la separación de las funciones de una aplicación (presentación, servidores de funcionalidad e información), ejemplos y los beneficios de esta Arquitectura.

En todo desarrollo de aplicaciones es muy importante el seguir una metodología, esto con el fin de apoyarnos en una herramienta que nos ayude al logro de los objetivos planteados; motivo por el cual el capítulo 4 "Implementación de Cliente/Servidor" inicia con el análisis de 11 pasos para el desarrollo de aplicaciones basadas en la Arquitectura Cliente/Servidor los cuales son: 1. Identificación de Funciones, 2. Identificación de Beneficios, 3. Funcionalidad y Fases correspondientes, 4. Consenso de Objetivos, 5. Determinación de Costos, 6. Determinación del Alcance, 7. Ubicación del Procesamiento de información, 8. Decisión de desarrollar o comprar componentes, 9. Estimación del Impacto, 10. Preparar a la Empresa para el Cambio, 11. Capacitación. El seguimiento de cada uno de éstos nos proporciona una metodología encaminada al éxito de la aplicación.

Así mismo, es necesario conocer qué herramientas existen en el mercado que cuenten con las características necesarias para el desarrollo de una aplicación con el fin de poder tomar la decisión de cuales elegir en cada caso; por tal motivo mostramos en este capítulo el conjunto de herramientas de software Ambiente Distribuido Abierto (ODE), mismas que son específicas para establecer un esquema Cliente/Servidor. Es importante señalar que en la actualidad las herramientas de la ODE se utilizan en todo el mundo y soportan la mayoría de plataformas y sistemas operativos.

Un punto de gran importancia a considerar en el desarrollo de una aplicación es: el sistema operativo bajo el cual se ejecutará ésta, sobre este tema se analizan las tres estructuras de sistema operativo que existen: la monolítica, la de capas y la de microkernel.

La parte final del capítulo está integrada por: a) los factores críticos en la implementación de Cliente/Servidor, específicamente tiempo de respuesta y transparencia. b) Beneficios de una Arquitectura Cliente/Servidor. En este apartado se analizan las ventajas del cliente, del servidor y en conjunto del esquema Cliente/Servidor (Performance, Escalabilidad y Plataforma Abierta).

En el capítulo 5 "Cajero Bancario", presentamos las partes fundamentales del análisis, diseño e implementación de una Arquitectura Cliente/Servidor en una sucursal bancaria, reduciendo dicha aplicación a las funciones de una caja de la misma. Consideramos que el sector bancario es un claro ejemplo de dónde se puede implementar una Arquitectura Cliente/Servidor por el gran volumen de datos que se maneja y por la gran diversidad de equipos y

de aplicaciones que tienen que integrarse sin sufrir modificaciones. Por otra parte ya que el sector bancario tiene un dinámico proceso de automatización Cliente/Servidor favorece en gran medida la independencia o transparencia de las aplicaciones con los medios de comunicación, permitiendo así incrementar el ancho de banda o de la tecnología de sus redes sin perturbar a las aplicaciones. La tendencia actual hacia los sistemas Cliente/Servidor favorecen el uso de las microcomputadoras con multiprocesadores (ALR Q-SMP) siendo éstas una solución a los añejos problemas de tiempo de acceso, portabilidad del software, conectividad y tolerancia a fallas. Cliente/Servidor incide en la mejor automatización de los procesos y en su actualización, sin modificar los datos que siempre se han manejado. Presentamos además gráficas basadas en instituciones financieras estadounidenses que muestran 1) que uno de los grandes problemas de la información en mainframe es la cantidad de tiempo requerido para contar con el acceso listo y produciendo; 2) el tiempo en que éstas planean moverse hacia una Arquitectura Cliente/Servidor, siendo más del 50% el que planea realizarlo en menos de 6 años, por lo que podemos observar la gran aceptación de ésta; y 3) el ascenso de sistemas basados en Cliente/Servidor donde más del 40% de las empresas ya lo utilizan. Como es lógico el sector Financiero Mexicano no se ha quedado atrás en la realización de sistemas basados en Cliente/Servidor, por lo que presentamos algunas de las instituciones que ya han implementado esta Arquitectura, así como algunos de sus sistemas realizados bajo este ambiente (Grupo Financiero Abaco, Bancomer, Multibanco Mercantil Probursa, Bancrecer Grupo Financiero, BANAMEX, entre otros). El diseño de la aplicación bancaria propuesta es una red Ethernet estrella 10base2 con un protocolo de comunicaciones TCP/IP, utilizando UTP par trenzado y conectores RJ45. Presentamos además el análisis y diseño de la misma, el diccionario de datos, los diagramas de flujo de datos, el diseño lógico tanto del cliente como del servidor, así como las herramientas utilizadas para la implementación de la misma, describiendo Sybase SQL y Visual Basic y la forma en que fueron utilizados en el desarrollo de la aplicación.

Como resultado del análisis de los capítulos anteriores exponemos las conclusiones a las que hemos llegado en el apartado correspondiente.

Además consta de tres apéndices formados de la siguiente manera:

Apéndice A en el que se incluyen los pasos técnicos seguidos para la configuración de la red.

En el Apéndice B presentamos el esquema de la base de datos generado para la aplicación en Sybase SQL Server. Aquí se incluye la creación de tablas, de índices, su tipo así como los permisos de acceso para cada uno de los usuarios.

En el Apéndice C se presenta la parte del código de Visual Basic que se generó para la máquina cliente. Este apéndice contiene únicamente las rutinas fundamentales del sistema, y que conforman la estructura del mismo.

En el Glosario incluimos la definición de los términos tanto informáticos como del negocio, que a lo largo de este seminario pudieran provocar la duda del lector en cuanto a su concepción y los cuales están resaltados en el texto con negritas itálicas.

La bibliografía está dividida en un apartado de libros y otro de revistas.

ANTECEDENTES

En las últimas cuatro décadas el tipo de procesamiento ha evolucionado de la siguiente manera :

En los sesentas los sistemas se ejecutan en ***mainframes*** dedicados al proceso de grandes volúmenes de datos, o procesos en batch.

En los setentas surgieron sistemas grandes y medianos (***minis***) de proceso centralizado con capacidad de soportar muchos usuarios a nivel departamental y corporativo por medio de tiempo compartido.

En los ochentas surge el concepto de la computadora personal (PC), con proceso local utilizando recursos propios para el procesamiento de información mejorando la productividad individual. También esta década es significativa por la introducción de poderosas ***estaciones de trabajo*** Unix, hechas con sistemas distribuidos y redes integradas, e incluso con procesadores ***RISC*** (Reduced Instructions Set Computing).

En los noventas una nueva clase de computación está conquistando la empresa de hoy y del futuro (pequeñas y grandes). La tecnología Cliente/Servidor de proceso distribuido debido a su arquitectura, que une los ambientes de cómputo anteriores, integra las tecnologías existentes (***mainframes***, ***minis***, PC's, macs, ***estaciones de trabajo***, supercomputadoras, etc.) para darles más utilidad, y proporciona a la empresa información precisa al nivel personal, departamental, empresarial, local y global.

EVOLUCION DE LA COMPUTACION DESDE EL MAINFRAME HASTA EL CLIENTE/SERVIDOR

La manera más sencilla de entender la importancia del concepto Cliente/Servidor es el comparar los modelos usados en ***mainframes*** y en la mayoría de las redes de PC antes de esta arquitectura.

1. Mainframe.

En el ambiente de ***mainframe***, las aplicaciones y los datos se encuentran almacenados en un máquina central (***host***) la cual provee la inteligencia para ejecutar la aplicación y procesar los datos, mientras que las terminales tontas conectadas a este ***mainframe*** se usan sólo para ver lo que se le solicitó y lo que éste está contestando. El host debe determinar cada responsable de las consultas de los usuarios, obtener la lógica del negocio y recuperar los datos (Figura A.1.).

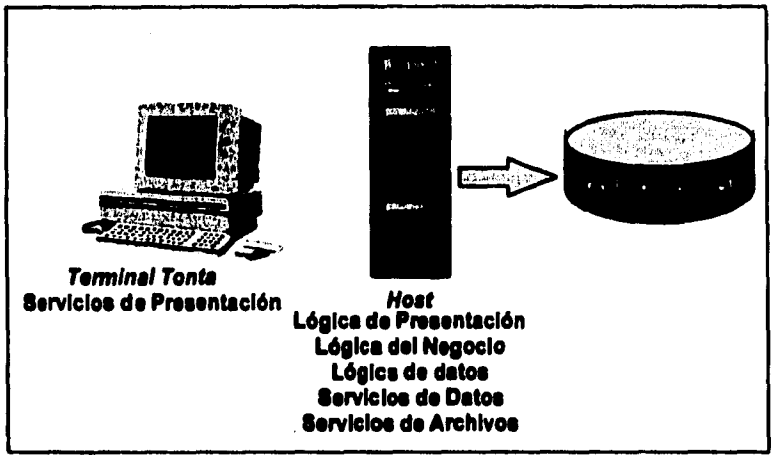


Figura A.1. Ambiente de Mainframe.

Este tipo de procesamiento ofrece las siguientes ventajas:

a) El poder de procesamiento del ***host*** proporciona un alto desempeño.

b) Como los datos se encuentran centralizados su integridad está bien protegida.

c) El ***mainframe*** utiliza programas y procedimientos altamente sofisticados para asegurar la confiabilidad.

Sin embargo este ambiente presenta desventajas importantes:

a) La interface de usuario está limitada al estilo de terminal para su interacción. Esto dificulta proveer una imagen en el ***GUI*** eficiente cuando todo el procesamiento inteligente reside en un ***host*** central. Desafortunadamente una terminal provee una interface no muy amigable para muchas aplicaciones.

b) Cada usuario y aplicación adicionales agregan una sustancial carga en el ***mainframe***, puede haber un incremento en el tráfico de la red más de lo necesario porque el ***host*** no sólo envía los datos a la terminal, además envía instrucciones de dónde se colocarán los datos en la pantalla, incluyendo etiquetas y prompts.

c) Cuando las necesidades de cómputo de la corporación aumentan, se requiere de una inversión de capital muy alta para agregar más poder de cómputo.

2. Servidor de Archivos.

En parte para responder a estas desventajas, las primeras generaciones de redes de área local fueron desarrolladas (***LAN***'s), en lugar de que el ***host*** efectuara todo el procesamiento de información se buscó utilizar la inteligencia de las ***estaciones de trabajo*** para ejecutar aplicaciones y el servidor simplemente tenía la función de permitir compartir archivos y periféricos a los usuarios de la red (Figura A.2.).

Ventajas:

a) Las PC's máquinas fáciles de usar funcionan como ***estaciones de trabajo*** .

b) El costo de una red de área local es considerablemente menor al de un ***mainframe*** o una minicomputadora.

c) Como el poder se encuentra en las ***estaciones de trabajo*** fácilmente se puede agregar más poder, integrando nuevas ***estaciones de trabajo.***

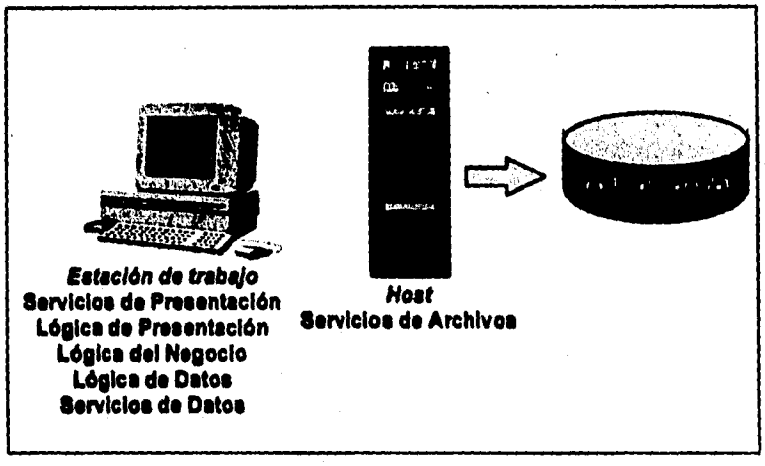


Figura A.2. Servidor de Archivos.

Desventajas:

a) Debido a que todo el proceso de información se efectúa en las ***estaciones de trabajo***, grandes volúmenes de información deben moverse de un sentido a otro de la red, degradando la velocidad de comunicación entre los equipos.

b) Como todos los usuarios tienen un acceso compartido a los mismos datos por medio de un servidor no inteligente, la integridad de los datos no es 100% confiable.

c) El control y administración de la red puede ser complicado, en especial conforme ésta crece.

3. Cliente/Servidor.

La Arquitectura Cliente/Servidor combina las ventajas de los modelos anteriores buscando minimizar y eliminar muchas de sus desventajas.

El concepto Cliente/Servidor trata tanto al servidor como a las ***estaciones de trabajo*** como dispositivos inteligentes y programables explotando todo el poder de cómputo de cada uno de éstos (Figura A.3.).

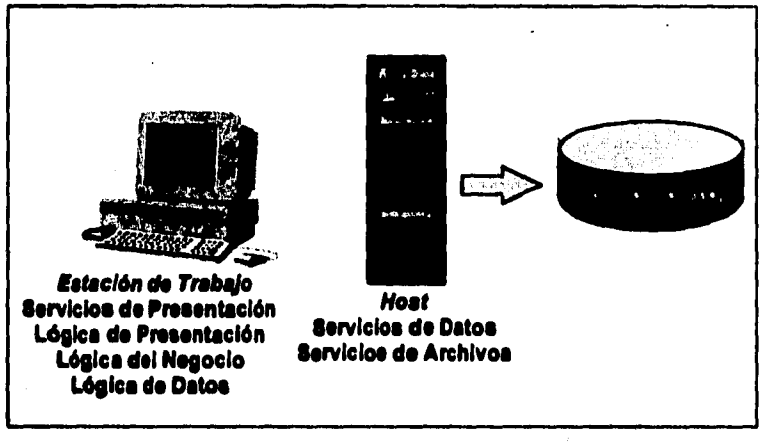


Figura A.3. Arquitectura Cliente/Servidor.

En los siguientes capítulos expondremos los fundamentos de Cliente/Servidor, sus elementos y cómo llevar a cabo la implementación de una arquitectura de este tipo.

1

AMBIENTE DISTRIBUIDO

- AMBIENTE DE COMPUTO DISTRIBUIDO (OSF/DCE)
- SISTEMA DISTRIBUIDO

Los estándares facilitan el funcionamiento de sistemas y organizaciones y el medio informático no es la excepción, más aun cuando existen elementos heterogéneos, los estándares son indispensables, ejemplos de estos hay muchos, el código ***ASCII*** o el código ***EBCDIC***. En una Arquitectura Cliente/Servidor los estándares en conjunto con las interfaces brindan la posibilidad de crear sistemas interoperables y portables.

Existen dos organizaciones de estándares que han tenido fuerte impacto en Cliente/Servidor, la ***X/Open*** y la ***OSF***.

La ***X/Open*** está formada por un consorcio de distribuidores de ***hardware*** y ***software*** que tratan de cubrir las necesidades del usuario final. La ***X/Open*** ha orientado sus esfuerzos para ofrecer a este usuario final un portafolio de especificaciones e interfaces que permitan la ***interoperabilidad*** y la portabilidad a nivel de código fuente, a este portafolio la ***X/Open*** le ha denominado CAE(Common Applications Environment).

La ***OSF*** tuvo sus comienzos con el establecimiento de un ambiente de plataformas para UNIX y hoy día se ha convertido en una de las principales organizaciones en cuanto a ambiente distribuido se refiere. La ***OSF*** fué fundada en 1987, los miembros originales de esta organización fueron Apollo Computer Inc., Digital Environment Corporation, Groupe Bull, Hewlett Packard, IBM, Nixdorf Computer AG, Philips y Siemens.

AMBIENTE DE COMPUTO DISTRIBUIDO (OSF/DCE)

El ambiente de cómputo distribuido (***OSF/DCE***) es un concepto desarrollado por la Open Software Foundation que abarca un conjunto integrado de sistemas operativos y de servicios independientes de la red que soportan el desarrollo, uso y mantenimiento de las aplicaciones distribuidas (Figura 1.1). Dada su capacidad para ofrecer una red administrable, transparente e interoperable de sistemas de diferentes plataformas y diversos fabricantes, ***DCE*** (Distributed Computing Environment) resulta ser una de las tecnologías más importantes de la década.

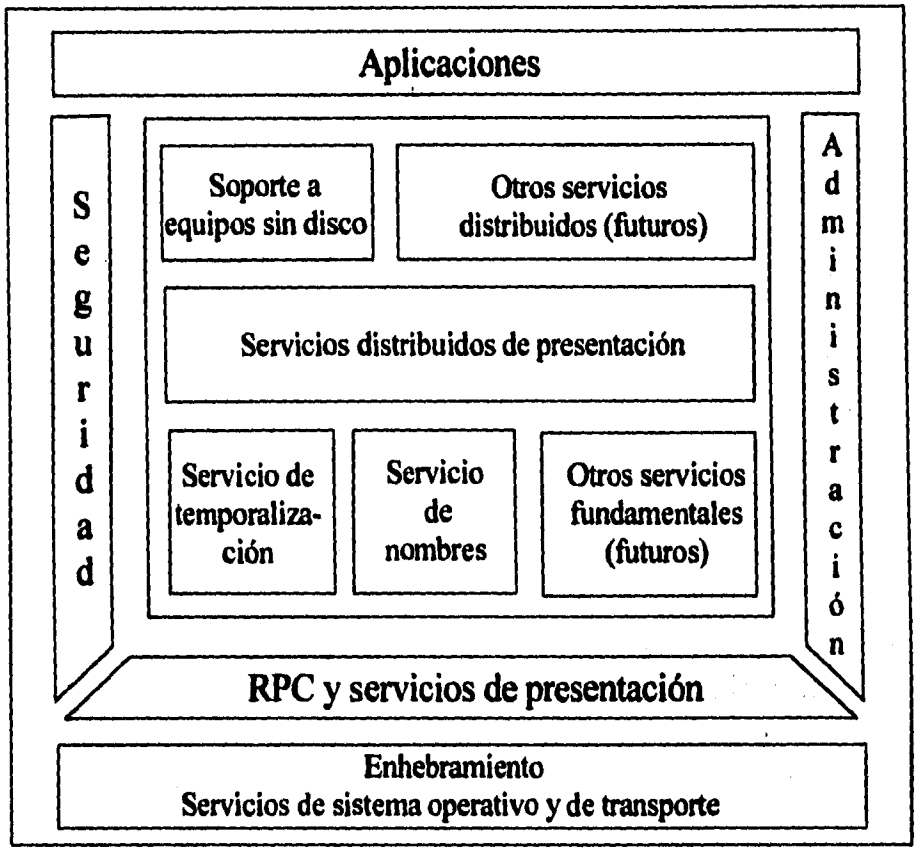
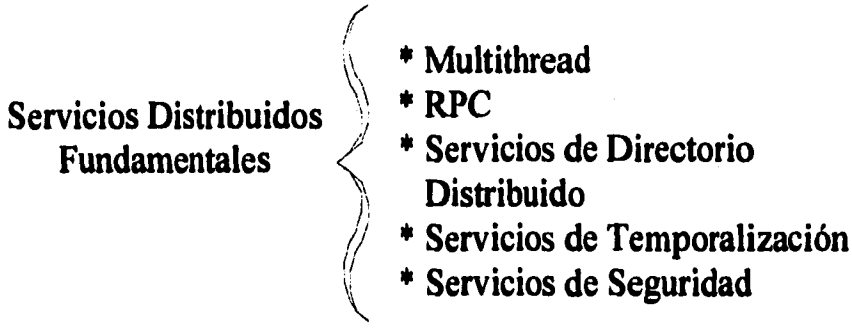


Figura 1.1. La Arquitectura del DCE de OSF

El ***DCE*** está construido sobre una arquitectura estratificada que integra un conjunto de tecnologías. Los prestadores de los servicios más elementales (como los sistemas operativos) están en la parte inferior, mientras que en el nivel más alto se encuentran los consumidores de servicios o aplicaciones. La seguridad y la administración son esenciales para todas las capas del modelo. Actualmente ***DCE*** consiste de siete herramientas y servicios que están divididos en servicios distribuidos fundamentales y servicios de compartición de datos.



Los servicios de compartición de Bases de Datos, construidos sobre los servicios fundamentales, incluyen:

Uno de los principales beneficios de ***DCE*** es que permite enmascarar la complejidad de un ambiente de red simplificando las labores de desarrolladores de aplicaciones, administradores de sistemas y usuarios finales; con este conjunto de servicios se logra cierto grado de independencia tanto de la red como del sistema operativo con cualquier plataforma y ***software*** de transporte incluyendo ***TCP/IP***, ***X.25*** y otros protocolos similares. Para el logro de tales objetivos, el conjunto de servicios de ***DCE*** deben estar dispersos a lo largo de toda la red.

A continuación explicaremos cada uno de los servicios distribuidos fundamentales.

Multithread. (Enhebramiento)

Tradicionalmente las aplicaciones manejan procesos, cada uno de los cuales tiene un "hilo" individual de control. En este modelo, las múltiples tareas incluidas en una aplicación se dividen entre múltiples procesos que se comunican.

Un programa con multithread queda descompuesto en múltiples hilos de ejecución. Los hilos son un modelo importante que está surgiendo para expresar el paralelismo dentro de un proceso, especialmente a través de un ambiente distribuido.

Por ejemplo, esta capacidad de multithread se vuelve particularmente importante dentro del contexto de un ***RPC***. Un ***RPC*** es síncrono por naturaleza: un cliente hace una llamada a una función remota y luego se queda esperando hasta que la llamada sea atendida. Sin embargo, un hilo puede hacer la petición, y otro puede comenzar a procesar los datos resultantes de diversas peticiones.

Remote Procedure Call. (RPC)

La idea básica de un ***RPC*** es el uso de llamadas a procedimientos remotos en un ambiente distribuido. Cuando un procedimiento remoto es invocado el mensaje contiene los argumentos necesarios para que se ejecute en una máquina remota donde se realiza el proceso y los resultados retornan a la máquina que realizó la llamada. Podemos decir que los ***RPC*** constituyen el mecanismo primario de comunicación entre los sistemas distribuidos, por su simpleza, familiaridad (porque es similar a una llamada local), en general y pueden ser implementados fácilmente en un ambiente distribuido. De esta forma podemos definir a un ***RPC*** como un mecanismo para llevar a la práctica el procesamiento distribuido que utiliza un modelo de programación muy conocido (la llamada a un procedimiento) a través de la red.

Un ***RPC*** de ***DCE*** ofrece sencillez. Se adhiere al modelo de procedimiento local tan cercanamente como es posible, proporcionando al

mismo tiempo los aspectos distribuidos de las aplicaciones en una manera directa.

La consistencia de protocolo es otra característica distintiva de un ***RPC*** de ***DCE***. El protocolo que utiliza un ***RPC*** está claramente especificado y no está sujeto a modificaciones por parte del usuario. Este núcleo garantizado es una consideración importante en un ambiente donde se requiere ***interoperabilidad***.

Sin importar el protocolo de transporte sobre el cual corra, un ***RPC*** de ***DCE*** proporciona una conducta idéntica y mantiene invisible el manejo de las conexiones. La interface de un ***RPC*** soporta una variedad de transportes simultáneamente, y permite la introducción de nuevos transportes y protocolos sin afectar la codificación de la aplicación.

Un ***RPC*** de ***DCE*** se integra con el sistema de autentificación (que explicaremos posteriormente) para brindar comunicaciones seguras. Se integra con hilos de cliente y de servidor, conservando la ***interface síncrona*** y al mismo tiempo permitiendo que tanto el cliente como el servidor aprovechen la concurrencia.

Mediante los ***RPC*** se puede simplificar la construcción de programas en un ambiente distribuido; para esto es indispensable construir los detalles de comunicación, la transmisión de errores y las fallas, lo ideal para esto es que las llamadas remotas conserven la misma semántica que las llamadas locales, por lo cual es fundamental tomar en cuenta aspectos como las diferencias en los tipos de datos y la necesidad de enviar datos en mensajes.

La interface ***RPC*** comprende varios niveles, en los niveles altos el desarrollador maneja servicios de ***RPC*** basados en librerías o implementa rutinas de sistemas simplificadas para ejecutar ***RPC***s. Por default las comunicaciones ***RPC*** y los ***sockets*** (de los cuales hablaremos en el capítulo 2) de protocolos son usados. En los niveles bajos el ***RPC*** puede ser utilizado para programación más flexible en la cual el desarrollador tiene control sobre la deliberación de mecanismos y ***sockets*** usados para transportar datos.

Un sistema de ***RPC*** debe proveer mecanismos para conectar la llamada a programas con la llamada a procedimientos. En el llamado a un programa está contenido el nombre del procedimiento que también será llamado, este nombre debe ser limitado al valor actual para el procedimiento, además de que es importante que el receptor y el emisor conserven una compatibilidad de interfaces o que su implementación esté de acuerdo a las tecnologías disponibles, de esta manera podremos lograr los alcances de los ***RPC*** con eficiencia como si se trataran de llamadas locales.

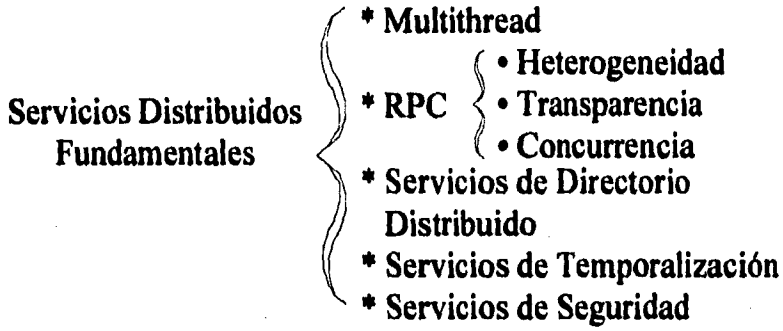
En un modelo distribuido de Cliente/Servidor los servidores manejan la recuperación de los datos y definen las operaciones para enviar una respuesta a los clientes; usando un manejador de bases de datos (***DBMS***, Data Base Management System), el acceso a datos se implementa con ***procedimientos almacenados*** (***RPC***) y/o por medio de mensajes.

Los ***procedimientos almacenados*** son soportados por cada proveedor de ***DBMS*** con su propio lenguaje, por ejemplo: Transact-SQL de Sybase, PL-SQL de Oracle. Los clientes invocan estas operaciones para manipular los datos manejados por el servidor, las operaciones son ejecutadas mediante interfaces de ***RPC***.

Algunos lenguajes de programación proveen soporte para procesamiento de ***transacciones***, típicamente en una aplicación es donde empieza una ***transacción*** que ejecuta uno o más ***RPC***'s a varios servidores y después finaliza ésta.

Un cliente puede ejecutar un ***RPC*** que invoque operaciones sobre un servidor que esté en el mismo ***nodo*** que el cliente, por otra parte la mayoría de los sistemas de ***RPC***'s son transparentes a la red, consecuentemente el cliente no necesita saber cuando el ***RPC*** va a través de la red ni tampoco necesita saber cuando la ***transacción*** es distribuida. Un servidor puede invocar mediante ***RPC*** a otros servidores o puede directamente leer y actualizar.

Características de un RPC



- **Heterogeneidad**

Actualmente los sistemas distribuidos exhiben una sustancial heterogeneidad, esto es, diferentes tipos de máquinas trabajan en una misma red, los programas que corren frecuentemente están escritos en diferentes lenguajes, además de que distintos sistemas operativos pueden correr en diferentes máquinas, de ahí surge un importante problema que consiste en cómo permitir la comunicación entre programas que corren en un ambiente heterogéneo, para esto es necesario definir la semántica de comunicación entre lenguajes y máquinas diferentes.

Muchos sistemas de ***RPC*** permiten la comunicación de programas sobre componentes heterogéneos mediante declaraciones estáticas de las interfaces de procedimientos remotos.

Una declaración de interfaces sirve para varios propósitos:

Primero - para que el que llama y el que es llamado tengan concordancia en el tipo de argumentos y resultados del procedimiento.

Segundo - la declaración de interfaces es básica para la revisión de tipos, verificando que la llamada así como el cuerpo de la misma estén correctas.

Tercero - La declaración de la interface puede ser usada para la generación automática de la conversión de código.

Una consideración importante al respecto es la conceptualización que guardan diferentes lenguajes sobre un mismo tipo de datos, para salvaguardar estas diferencias los tipos usados en la interface para las llamadas a procedimientos remotos deben ser definidas independientemente de cualquier lenguaje de programación. El problema de la conversión entre diferentes representaciones puede ser solucionado por varios caminos como definir una representación estándar para cada tipo de mensaje y que es requerida para cada emisor y receptor.

- **Transparencia**

Un gran éxito de muchos sistemas de ***RPC*** es el hacer que la semántica de las llamadas remotas sean o funcionen como en una llamada local, haciendo muy difícil distinguir entre una y otra su funcionamiento. El principal punto en la transparencia es la semántica y el paso de parámetros en las llamadas remotas.

- **Concurrencia**

En algunos sistemas de ***RPC*** la llamada es suspendida hasta que el procedimiento retorna un resultado, esto puede ocasionar problemas si los clientes o los servidores están conectados por una sola vía. Por ejemplo un cliente puede atender otro trabajo mientras el ***RPC*** es ejecutado en el servidor. También un servidor puede hacer llamadas a otros servidores o puede brindar servicio a otros clientes mientras espera las respuestas de las sub-llamadas.

Una forma de aligerar el proceso es que el sistema de ***RPC*** trabaje en bloques. Así un servidor puede correr llamadas concurrentes de múltiples clientes cada una separada en procesos. Similarmente un cliente puede crear varios procesos para realizar trabajos concurrentes.

Servicios de Directorio Distribuido

Localizar cosas (como usuarios, recursos, datos o aplicaciones) en una red distribuida es la tarea del servicio de directorio. El nombre o los servicios de directorio deben mapear grandes cantidades de objetos del sistema (usuarios, organizaciones, grupos, computadoras, impresoras, archivos, procesos y servicios) con nombres orientados a objetos.

Existen cuatro elementos en el servicio de directorio de ***DCE***:

Servicios de Directorio de Celdas (CDS).

Una celda de la red es un grupo de sistemas administrado como una sola entidad. El ***CDS*** está optimizado para el acceso local. El grueso de las consultas para el servicio de directorio se refiere a recursos que están dentro de la misma celda del originador. Cada celda de la red necesita por lo menos de un ***CDS***.

***Agentes de Directorio Global* (GDA).**

El ***GDA*** es una compuerta de nomenclaturas que conecta el dominio de ***DCE*** con otros dominios administrativos a través del servicio de directorio a nivel mundial ***X.500*** y de ***DNS*** (Servicio de nombre de Dominio). El ***GDA*** lleva las consultas de los nombres a los que no puede encontrar en la celda local y se los pasa a otro servicio de celda o al servicio de directorio global (Dependiendo de la ubicación del nombre). Para buscar un nombre, un cliente consulta al ***GDA*** local. El ***GDA*** pasa entonces una consulta de nombre inter-dominio al servicio ***X.500***. Este servicio devuelve la respuesta al ***GDA***, el cual a su vez se lo envía al cliente.

***Servicio de Directorio Global* (GDS).**

Basado en el ***estándar X.500***, el ***GDS*** funciona como un nivel más alto de jerarquía de directorios con el fin de conectar múltiples celdas en varias organizaciones.

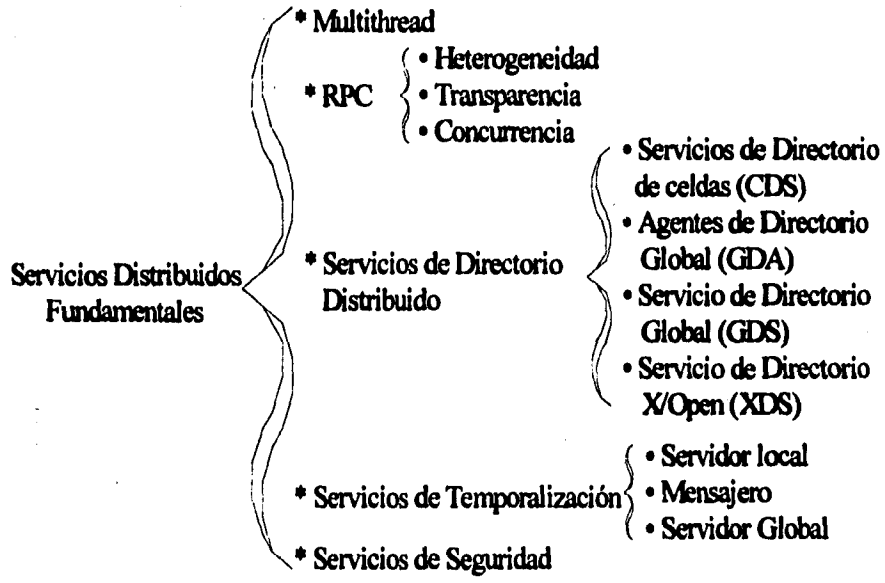
***Servicio de Directorio* X/Open (XDS).**

El soporte a las llamadas de servicio de directorio a la ***API*** de ***X/Open*** permite crear aplicaciones que sean independientes de la arquitectura subyacente de servicio de directorio. Una aplicación que se apegue a ***XDS*** funcionará sin necesidad de modificación con los servicios de directorio de ***DCE*** y de ***X.500***.

Servicios Distribuidos de Temporalización

Los servicios de redes distribuidas necesitan un servicio de temporalización consistente. Muchos servicios distribuidos, tales como los sistemas distribuidos de archivos y los servicios de autentificación, comparan las fechas generadas en computadoras diferentes. Para que la comparación tenga sentido, ***DCE*** debe mantener una marca de tiempo

consistente, cada máquina es responsable de verificar el tiempo actual y ajustar su reloj local.



Los Servicios Distribuidos de Temporalización (***DTS***) usan tres tipos de servidores para coordinar el tiempo en la red. Un servidor local sincroniza con los otros servidores de la misma ***LAN*** extendida o de una ***WAN***. Un mensajero es un servidor local designado que se coordina regularmente con los servidores globales. Los servidores pueden obtener la hora universal coordinada oficial a partir de organizaciones de estándares vía radio de onda corta, líneas telefónicas o por satélite.

Servicios Distribuidos de Seguridad

Existen dos amplias categorías generales de seguridad:

La autentificación verifica la identidad de una entidad (usuario o servicio).

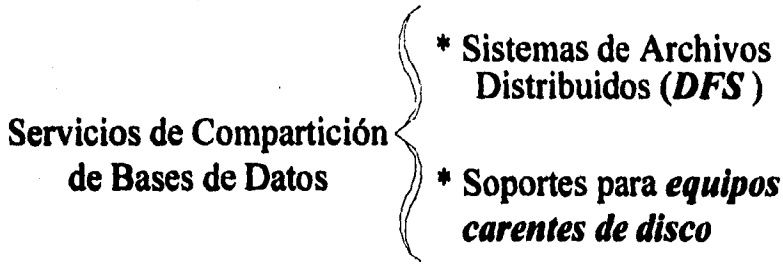
La autorización (o control de acceso) concede privilegios a la entidad, tales como acceso a un archivo.

Sin embargo, la autorización por sí misma es sólo una solución parcial. Los servicios de autentificación deben existir para un ambiente distribuido de red en donde no se puede confiar en que una ***estación de trabajo*** se identifique correctamente a sí misma o a sus usuarios para compartir servicios de la red.

La seguridad de la ***OSF*** está basada en el sistema de autentificación ***Kerberos***, y está aumentando con componentes de seguridad. ***Kerberos*** usa ***encriptamiento*** de llave privada para proporcionar tres niveles de protección. El nivel más bajo sólo requiere que la autenticidad del usuario sea establecida al inicio de una conexión, asumiendo que los mensajes subsecuentes para la red provienen del principal ya autentificado. El siguiente nivel requiere de la autentificación de cada mensaje de la red. Un

nivel más allá de estos mensajes seguros se encuentran los mensajes privados, en donde cada mensaje es ***encriptado***, además de autentificado.

La ***OSF*** agregó también un servicio de registro y un servicio de autorización. También incluye verificaciones de autorización basadas en las listas de control de acceso y una interface de autentificación para un ***RPC***.



Sistemas de Archivos Distribuidos (*DFS*)

Un Sistema de Archivos Distribuidos (***DFS***) permite brindar a los usuarios la percepción de que los archivos están localmente alojados en los lugares donde se llevan a cabo las ***transacciones***, esta percepción se logra gracias a un sistema operativo de red que es capaz de redireccionar de forma local o remota los requerimientos de los usuarios a través de los servidores que accesan a los archivos que contienen los datos requeridos. En contraparte a la ventaja que representa poder compartir archivos iguales entre varios usuarios a un mismo tiempo, está la posibilidad de consultas de datos erróneas, para lo cual se debe contar por medio del sistema operativo con adecuados mecanismos de control de actualizaciones y flujo de datos.

Por ejemplo, un sistema distribuido de archivos debe tener un espacio uniforme de nombres. Los archivos deben tener el mismo nombre, sin importar su plataforma y su ubicación. Otras facilidades a considerar son la seguridad, la consistencia de datos y la disponibilidad integradas; la confiabilidad y la recuperación, el desempeño y la escalabilidad para configuraciones muy grandes sin sufrir degradación, el manejo y la administración coherentes con independencia de la ubicación.

La puesta en práctica de ***DFS*** proporciona un excelente ejemplo de cómo los diversos componentes de ***DCE*** trabajan juntos. El ***software*** de ***DFS*** reside en cada nodo de la red. ***DFS*** integra el sistema de archivos del nodo con los servicios de directorio de ***DCE***, asegurando una convención uniforme de nombres para todos los archivos almacenados bajo ***DFS***.

Para maximizar el desempeño del acceso a archivos, ***DFS*** pone en un ***cache*** los archivos que son usados frecuentemente en un disco local de las ***estaciones de trabajo***. Cuando un usuario hace acceso a datos del servidor de archivos (***file server***), una copia de los datos son enviados a un ***cache*** localmente. Cuando el usuario termina de trabajar con los datos, el archivo vuelve a ser escrito sobre el servidor. El resultado es que el usuario puede tener acceso con rapidez a archivos distribuidos.

Para evitar que surjan problemas cuando múltiples usuarios en diferentes computadoras tienen acceso a los mismos datos y los modifican, ***DFS*** usa un esquema de manejo de estafeta para coordinar la modificación del archivo. Esto evita la corrupción accidental de los archivos distribuidos causada por actualizaciones no sincronizadas.

DFS permite a los administradores de sistemas subdividir particiones del sistema de archivos en filesets (Colecciones lógicas de archivos). Los filesets no están montados en el espacio de nombres del sistema de archivos local, sino que están empalmados en el espacio de nombres de directorio global de ***DCE***. Las referencias a un fileset se hacen por medio de su nombre en el directorio global, así que su nombre es independiente de su ubicación.

De esta forma, los filesets facilitan la administración. Si la partición de un disco está a punto de saturarse, el administrador puede pasar los filesets a otra partición o a otro servidor de archivos.

SISTEMA DISTRIBUIDO

"Un sistema distribuido es un sistema con elementos de procesamiento y dispositivos de almacenamiento, que se conjuntan en una red"[1]. Esto hace a los sistemas distribuidos más eficientes que los centralizados, por dos principales razones:

Primera - la tolerancia a fallas, cuando un procesador falla, otro puede realizar el trabajo gracias a que los recursos de la computadora pueden estar dispersos en diferentes equipos, si uno falla el otro entra en función en apoyo al que falló.

Segunda - un sistema distribuido permite trabajar en paralelo debido a la facilidad para compartir recursos y llevar a cabo múltiples tareas a la vez.

Así mismo, consideramos importante la construcción de sistemas distribuidos por las siguientes razones:

- Facilidad de comunicar y compartir recursos e información
- Costo/Rendimiento
- Modularidad
- Escalabilidad

Tabla 1.1 . Razones para construir Sistemas Distribuidos

- **La facilidad de comunicar y compartir recursos e información** por parte de distintos elementos de procesamiento, esto permite que información generada en un lugar y que es necesaria en otro, sea compartida.

[1]Distributed systems. Sape Mullender. Editorial ACM Press Frontier Series. 1991.

- **Relación Costo/Rendimiento.** "El costo de una computadora depende del grado de rendimiento alcanzado en términos de velocidad de procesamiento y de capacidad de memoria. El costo de estos elementos generalmente va en decremento, sin embargo, el costo de la comunicación depende del ***ancho de banda*** del ***canal*** y de la longitud del mismo. El ***ancho de banda*** se incrementa pero no más allá de los límites de los cables y las interfaces usadas, los cables para conectar redes lejanas han sido usados durante décadas por que ha sido lo más económico y un cambio representa un alto incremento en el costo"[2]. Al ser más poderosas las computadoras en términos de resolución de gráficos, video y voz la demanda en términos de rendimiento de las comunicaciones también a ido en aumento. Estos efectos hacen que los sistemas distribuidos no sólo sean económicos sino necesarios para satisfacer los requerimientos de usuarios que exigen sistemas amigables, con interfaces gráficas y presentaciones agradables, las cuales sólo se pueden lograr con poderosos equipos de escritorio que trabajen en un sistema distribuido.

- **Modularidad.** La división de los procesos en módulos permite que sean ejecutados en diferentes equipos o bien en diferentes procesadores. Un elemento fundamental en la comunicación entre estos módulos son los ***RPC***.

- **Escalabilidad.** "Un sistema distribuido es capaz de crecer, es decir de incrementar dispositivos de almacenamiento o unidades de procesamiento"[3], esto debido a la flexibilidad que ofrece su diseño.

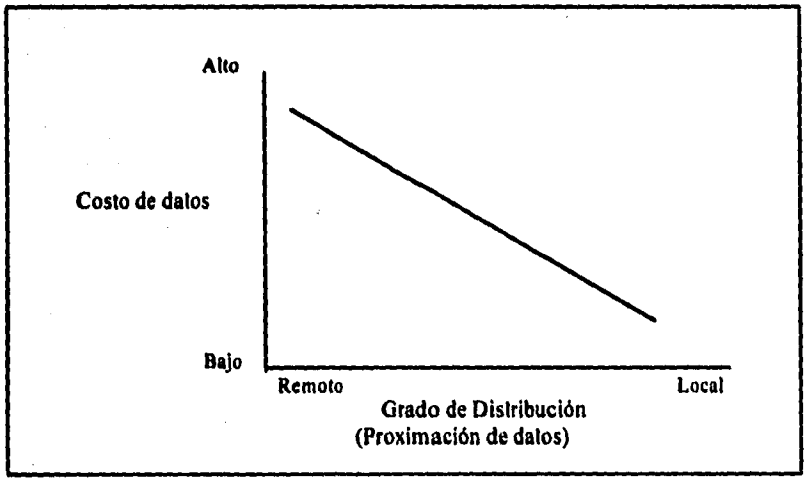
Distribución de Datos

Uno de los conceptos importantes dentro de la Arquitectura Cliente/Servidor es el de distribución de datos. Podemos decir que en sí constituye una de las principales ventajas de la Arquitectura Cliente/Servidor, pues permite que los datos puedan ser accesados desde

---

[2] Idem.
[3] Idem .

estaciones locales o remotas por medio de servidores y a través de un ***ancho de banda***. Precisamente este punto constituye uno de los principales fundamentos para construir modelos Cliente/Servidor en vez de los tradicionales esquemas centralizados. "Elbert y Martyna" [4] nos explican las ventajas de la distribución de datos en términos de costo; donde nos exponen que el costo del acceso a los datos está directamente relacionado con el grado óptimo de ejecución de una aplicación y la viabilidad que existe sobre el acceso a los datos, del argumento de los autores antes mencionados se puede concluir que el costo de los datos se reduce según el grado de distribución de los mismos. (Ver gráfica 1.1)



Gráfica 1.1. Distribución de datos Vs. Costo de Datos

En esta gráfica podemos observar cómo el costo de ejecución se va reduciendo a medida que los nodos de residencia de datos va aumentando, debido a la proximidad de los datos al lugar de procesamiento y consulta de los mismos. Es por eso que en el diseño e implementación de una Arquitectura Cliente/Servidor tenemos que considerar la posibilidad de situar a los usuarios en los nodos de residencia de datos más próximos geográficamente, como se muestra en la figura 1.2.

[4]Client/Server Computing Architecture, Applications, and Distributed Systems Management. Bruce Elbert, Bobby Martyna Editorial Artech House, inc. 1994. pp. 26.

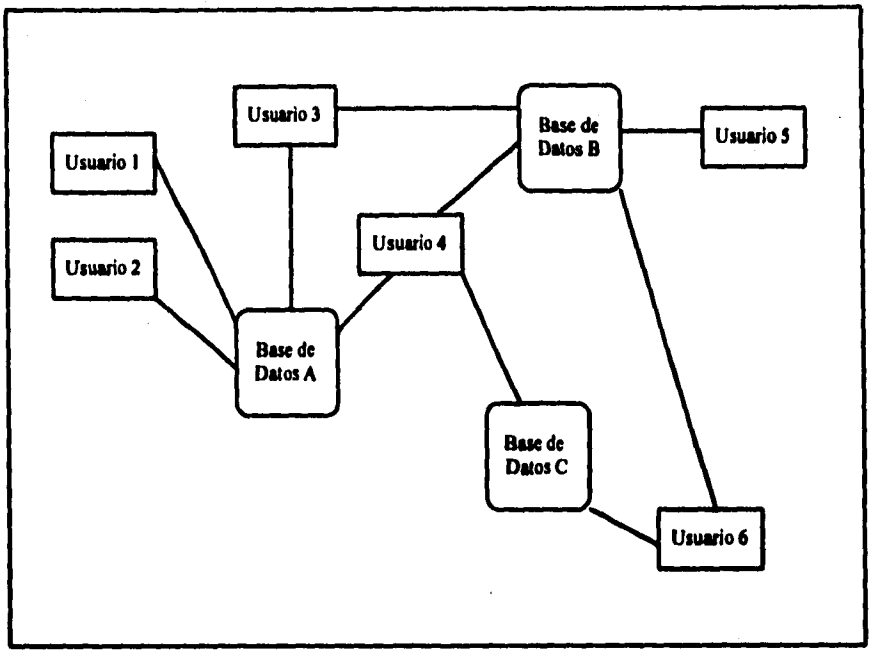


Figura 1.2. Distribución y Acceso de Datos

Sin embargo este concepto de distribución de datos no puede desligarse de la importancia de distribuir también de forma adecuada las unidades de procesamiento. "Cliente/Servidor trata de explotar al máximo las capacidades del procesamiento distribuido a través del óptimo equilibrio en los costos de realización entre las plataformas existentes"[5]. Este concepto involucra el llevar a cabo labores de ***downsizing*** si es que se tuviera todo el procesamiento concentrado sobre una plataforma de mainframe.

"En el procesamiento distribuido el proceso de funciones es distribuido sobre 2 o más sistemas de cómputo independientes, así de este modo los recursos de más de una computadora están disponibles para cubrir tareas que se requieran fuera de ésta"[6].

[5]Idem 4. pp 28.
[6]Idem 5. pp 28.

Los puntos anteriormente mencionados involucran integrar varios aspectos en una Arquitectura Cliente/Servidor tales como el contar con adecuados sistemas de archivos distribuidos, de bases de datos distribuidas, de correo electrónico y de intercambio electrónico de datos.

2

COMPONENTES DE CLIENTE/SERVIDOR

- FRONT END - BACK END
- SERVIDOR
- CLIENTE
- PROTOCOLOS DE COMUNICACION
- FACTORES CRITICOS EN LA IMPLEMENTACION DE CLIENTE/SERVIDOR

La Arquitectura Cliente/Servidor es una alternativa basada en la distribución del procesamiento a través de equipos que están conectados en redes. Este modelo distribuye las funciones de procesamiento de información así como los recursos de la misma, tales como sistemas manejadores de bases de datos, herramientas de escritorio y aplicaciones, todo sobre múltiples plataformas.

En la Arquitectura Cliente/Servidor, las máquinas clientes ejecutan la lógica de procesamiento de despliegue. Los servidores de base de datos almacenan información, manejan las tareas de almacenamiento y recuperación de la misma, en tanto que los servidores de comunicaciones o bien los mismos equipos manejan las funciones de ruteo de paquetes a través de la red. Esta arquitectura separa las ***aplicaciones monolíticas*** en componentes individuales que interoperan unos con otros sobre una red para cubrir los requerimientos de los usuarios.

FRONT END - BACK END

Cliente/Servidor contempla a nivel de funciones dos elementos, una llamada **Front End,** y la otra **Back End.**

Front End : Es la parte de la aplicación que interactúa con el usuario, está ubicada en la máquina cliente; está orientada a la interpretación y presentación de la información procedente del servidor, generalmente está conformada bajo un ambiente gráfico, de ahí que recalquemos la importancia de la capacidad del cliente para el manejo de ***GUI's.***

Back End : Son los módulos que residen en el servidor, básicamente orientados a recibir solicitudes del cliente con capacidad de procesamiento y de regresar la información solicitada a éste.

Hay que destacar que esta arquitectura se encuentra particularmente provista para el manejo distribuido de datos, debido a que rompe las aplicaciones lógicamente para lograr un procesamiento más rápido y eficiente.

Los Sistemas Manejadores de Bases de Datos Relacionales (DBMS) juegan un rol muy importante en la Arquitectura Cliente/Servidor ya que aseguran la integridad de la información y la seguridad de la misma, a la vez que proveen el control y la disponibilidad necesarios para aplicaciones de producción en línea. Así mismo, los DBMS manejan la recuperación de los datos.

Para comprender mejor la Arquitectura Cliente/Servidor a continuación hablaremos de sus componentes generales (Servidor, Cliente, Protocolos) y su funcionamiento.

SERVIDOR

Consideramos al Servidor como una computadora que provee a los clientes de elementos como discos de gran capacidad, bases de datos y/o conecciones a la red. Es importante señalar que hablamos de servidores inteligentes. Los servidores pueden ser mainframes, minicomputadoras o potentes ***estaciones de trabajo***.

En el ***Back-End*** las aplicaciones efectúan actividades que requieren un consumo importante de recursos de procesamiento, mejorando ampliamente el poder de proceso de la red. Explotando la inteligencia del servidor, se puede contar con una administración de datos más controlada, una mejor seguridad y una más fácil administración de la red. **La diferencia de un Mainframe es que en esta Arquitectura Cliente/Servidor la interface del usuario, así como otros procesos específicos de la aplicación, se efectúan del lado del cliente.**

El servidor también debe efectuar toda la labor intensiva de comunicación con otros ambientes, redes y topologías.

Categorías de Servidores

A continuación se describen las diferentes funciones que pueden ser ejecutadas por los equipos servidores.

• **Servidor de archivos.**
• ***Servidores de Impresión.***
• **Servidores de Comunicaciones.**
• **Servidor de Base de Datos.**
• **Servidores de *Fax*.**
• **Servidores de Correo.**

Tabla 2.1. Categorías de Servidores

- **Servidor de Archivos.**

El Cliente maneja toda la presentación, la lógica del negocio y las funciones de la base de datos. El servidor de archivos se encarga de enviar una copia de los archivos necesarios y los envía a través del cliente. Los servidores de archivos atienden los requerimientos de archivos por parte de los usuarios o de las estaciones de trabajo. Estos requerimientos son sólo para archivos individuales, el servidor no manipula el contenido de éstos.

Para crear mejores ***Interfaces Gráficas de Usuario (GUI's)*** de una aplicación fácilmente, basta una computadora cliente para manejar todas las funciones de despliegue, evitando el retraso de la red entre la lógica de presentación y los servicios. Por otro lado la escalabilidad es mejor, debido a que los usuarios y las aplicaciones solamente se agregan incrementando sólo un poco la carga del ***CPU***, mientras que el servidor de archivos sólo recupera los mismos. Cabe señalar que cada usuario adicional brinda un poder de procesamiento extra para la red.

Sin embargo, a pesar de esto la arquitectura Servidor de Archivos tiene dos desventajas significantes. Primero la demanda de las computadoras personales es muy elevada. Por ejemplo, una interface basada en Microsoft Windows, que ya representa un código complejo para la aplicación, y un ***DBMS*** en una misma máquina puede degradar el rendimiento de una PC. Segundo, y más importante, es el gran incremento de la carga de la red.

Esto se mostrará con el siguiente ejemplo: Supongamos que se tiene una orden de entrada a la base de datos que contiene una tabla de clientes, una tabla de órdenes, una de partes y una tabla de campos la cual liga las tablas de partes y órdenes. Un requerimiento razonable debiera ser un listado de cada cliente, cada parte de lo ordenado por el cliente y el valor total de cada parte ordenada. En un sistema de servidor de archivos esto requiere mover todo el contenido de las cuatro tablas a través de la red hasta el cliente, descuidando qué tantos datos se involucran. La siguiente figura representa lo anterior gráficamente :

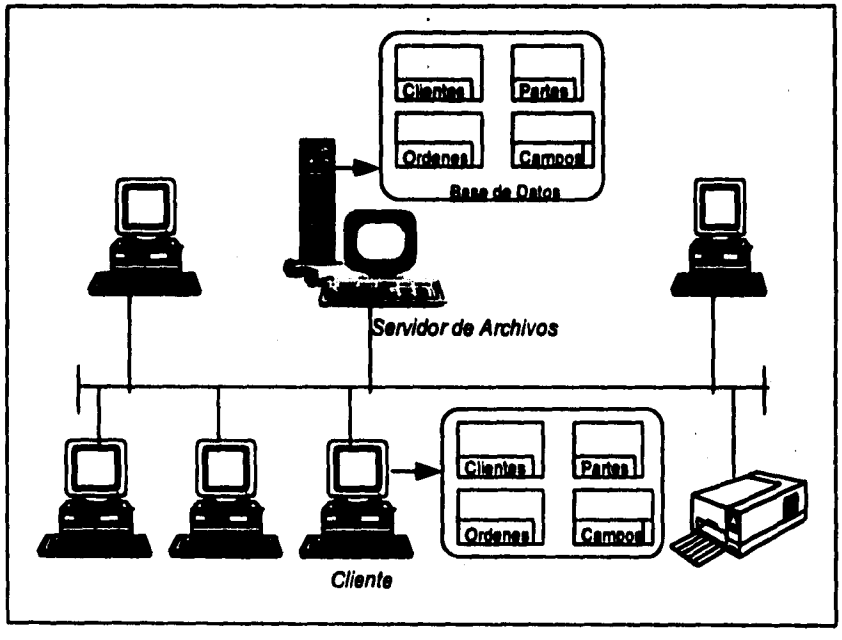


Figura 2.1. Ejemplo de Saturación de la Red

El concepto Cliente/Servidor esta diseñado para manejar estos problemas por la separación de las partes componentes de una aplicación, las cuales trataremos posteriormente, y su lugar (dónde es más efectivo que estén). Debido a que existen diferentes divisiones es necesario entender las diferentes arquitecturas y para qué tipo de aplicaciones es apropiada cada una.

- **Servidor de Base de Datos.**

El Servidor de Base de Datos debe asegurar la ***integridad de los datos*** dentro de la misma base con objeto de que las políticas de la empresa en cuanto a afectación de la información sean aseguradas en todas las aplicaciones de la red. Tanto la administración como la seguridad son mantenidas en el servidor para asegurar la consistencia a través de todas las aplicaciones.

Consideraciones para la Implementación de Servidores.

- Debe hacerse un análisis previo de las aplicaciones existentes, de los requerimientos del usuario, del tamaño y los recursos con que cuenta la empresa.

- Determinar cuántos servidores y clientes son necesarios para soportar las aplicaciones que está requiriendo la empresa.

- En qué lugares se necesitan establecer estos clientes y estos servidores.

- Establecer máximos y mínimos de soporte.

Elbert y Martyna también nos presentan cuatro alternativas primordiales de ***hardware*** que podemos emplear para los servidores :

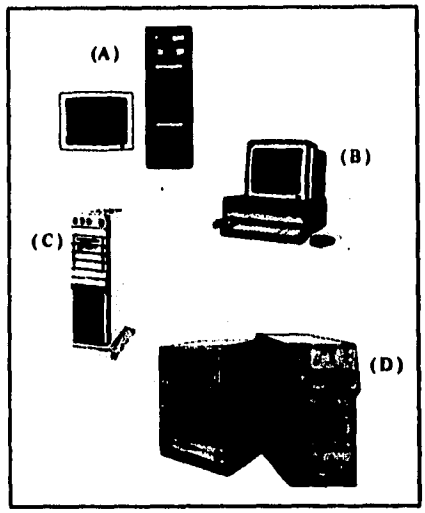


Figura 2.2. Alternativas de computadoras para Servidores (A)High-end PC *o Macintosh; (B)Unix estación de trabajo o RISC; (C)Sistema Midrage, Microvax, AS400 ó Multiprocesador Simétrico (SMP) Super Servidor; (D)Mainframes. Cualquiera de éstos puede ser accesado por múltiples clientes en la red.*

En la llamada Arquitectura de Servidor de Archivos todo el procesamiento de aplicaciones ocurre en el cliente, y el servidor sólo cubre los requerimientos que estos le soliciten. En esta arquitectura los usuarios pueden requerir el acceso a los archivos del servidor de archivos y el servidor está conectado a la estación de trabajo, donde el usuario procesa los datos de la forma en que éste desee.

Los sistemas basados en Servidores de Archivos tienen algunas limitaciones, una de éstas es que múltiples usuarios están imposibilitados de actualizar los mismos datos simultáneamente. Los archivos pueden ser repartidos sólo por una máquina en un tiempo. Otro problema es que la ***estación de trabajo*** es requerida para todo el procesamiento de ésta misma lo cual restringe la cantidad de datos que pueden ser procesados, hay que recordar que no solamente es la ***estación de trabajo*** la responsable de soportar sus propias aplicaciones sino también de soportar el procesamiento de la red. Esto representa un problema porque los servidores de archivos no

tienen inteligencia, éstos responden a los requerimientos de datos con el envío de archivos enteros a través de la red, esto necesariamente incrementa el tráfico de la misma involucrando problemas con el ***ancho de banda*** con que se cuente.

Por ejemplo: si una ***estación de trabajo*** está procesando la consulta de un usuario acerca de "todos los clientes que viven en el Distrito Federal", este requerimiento es enviado al servidor de archivos el cual a su vez enviará todo el archivo de clientes de vuelta a la ***estación de trabajo*** quien ejecutará la consulta y en la cual sólo aparecerán 5 clientes; esto en una red que cuente con un número mínimo de ***nodos*** no representará mayor problema sin embargo nos podemos imaginar la degradación en tiempo de respuesta y el congestionamiento causado en redes con gran número de ***nodos***.

A través de una Arquitectura Cliente/Servidor se pueden aprovechar y combinar las capacidades de equipos heterogéneos como una PC, una ***estación de trabajo,*** una ***minicomputadora*** o un ***mainframe***, este esquema constituye la siguiente etapa de la computación basada en clientes.

La Arquitectura Cliente/Servidor es una forma de proceso distribuido, pero que toma el concepto del modelo basado en clientes y lo lleva más lejos. En este modelo un servidor es responsable solamente de repartir archivos y recursos de periféricos, así el servidor puede actuar como una computadora corriendo aplicaciones en red, ésto hace posible que en la Arquitectura Cliente/Servidor se distribuya el procesamiento de funciones entre las estaciones de trabajo y los servidores de una manera mucho más eficiente, así un servidor no sólo puede retornar ***archivos planos*** sino que puede llevar a cabo labores de ordenamiento, selección e indexación.

En esta arquitectura el servidor opera como un ente inteligente capaz de manejar los requerimientos de los clientes y sólo retornar los valores de las consultas hechas funcionando así como un servidor de bases de datos.

El desarrollo del servidor ha dado lugar a las siguientes ventajas :

- Habilidad para recuperar y almacenar grandes cantidades de datos
- Habilidad para asegurar la integridad del almacenamiento de datos
- La separación del almacenamiento físico y la presentación de datos, permitiendo a la base de datos ser actualizada sin impactar las aplicaciones que accesan a la base (independencia de datos)
- La habilidad para presentar múltiples vistas de datos como soporte de múltiples departamentos, aplicaciones y usuarios
- Soporte de administración de datos para controlar el acceso a los mismos
- Soporte de un lenguaje estándar SQL
- Habilidad para procesar cientos o tal vez miles de transacciones por segundo

Tabla 2.2. Ventajas del Desarrollo del Server

La siguiente tabla muestra algunos de los proveedores que han apoyado la Arquitectura Cliente/Servidor :

Proveedor	Modelos Propuestos	Características Principales
Compaq Computer de México	-Línea ProSignia (uso general, donde la caída del servidor no es crucial para la empresa): -ProLaint (misión crítica) : VS, 1000, 2000, 4000	ProLaint : * Monitoreo de la Temperatura * Error Correcting Code (ECC) (notifica la falla, aisla la zona y corrige errores en el disco) * RAID 1, 4, 5, 10 (el servidor sólo se abre para retirar el disco dañado y colocar el nuevo) * SmartStart CD (posee todas las versiones actuales de los principales sistemas operativos de red en CD, permitiendo configurar la máquina en 30 minutos de forma óptima) * Automatic Server Recovery (permite al equipo arrancar automáticamente después de una falla para llegar al mismo punto donde se cayó el servidor) * Pager (Receptor de mensajes Individual, avisa de algún problema)

Tabla 2.3.1. Proveedores importantes para Servidores [1]

[1] PC Magazine en español, Vol. 5 Núm. 6 ,1994, pp 86-88

Proveedor	Modelos Propuestos	Características Principales
Unisys de México	-Advantage Plus MPE5606 y MPE5608 -Servidores de la familia PW	* Procesador Pentium a 60 MHz, memoria base de 8MB expandible a 192MB en tarjeta principal, memoria caché de 256K, controlador de vídeo ***EVGA*** con 1 MB de ***VRAM***, controlador de disco duro con conector externo, seis puertos disponibles para el primer modelo y ocho para el segundo * Están diseñados para aceptar las nuevas unidades de floppy como la de 2.82MB sin tener que cambiar la tarjeta (posibilidad de actualización en futuras tecnologías)
Hewlett-Packard	-Familia NetServer series : LF y LM	* Soportan grandes capacidades de almacenamiento como bases de datos * 384MB de memoria, opcionalmente memoria de detección y corrección de errores * HP protege la inversión del cliente, tanto en memoria, discos o procesadores sin la necesidad de cambiar todo el equipo

Tabla 2.3.2. Proveedores importantes para Servidores [2]

[2] PC Magazine en español, Vol. 5 Núm. 6, 1994, pp 86-88

Proveedor	Modelos Propuestos	Características Principales
Dell Computer de México	- Dell PowerEdge : (aplicaciones de grupo en redes pequeñas o medianas) XE (aplicaciones de organizaciones medianas o grandes)	* Sistemas basados en procesador i486 de Intel * Incluyen tecnología de bus local Peripheral Component Interconect (PCI), subsistema de disco Dell de alto rendimiento * Monitoreo térmico para verificar el comportamiento de la temperatura del equipo. * Memoria ECC (Error Correcting Code)

Tabla 2.3.3. Proveedores importantes para Servidores [3]

[3] PC Magazine en español, Vol. 5 Núm. 6, 1994, pp 86-88

CLIENTE

Hemos considerado al cliente como una computadora, ya sea una ***mini***, PC o ***estación de trabajo*** conectada a una red que se utiliza para accesar información y recursos de la red, así como sus propios recursos. Los programas del lado del cliente representan una parte del total de la aplicación que le permite al usuario el interactuar con la misma. Esto típicamente es una interface que le permite iniciar la aplicación, solicitar datos y desplegar información. Para el usuario debe verse como si estuviera trabajando en una computadora *stand-alone*.

La necesidad de más poder de las herramientas desarrolladas para los clientes ha crecido tanto como el ***hardware*** ha evolucionado, el cliente fue típicamente una terminal o una PC 8088 con 256 kilobytes de memoria y tal vez 10 megabytes de almacenamiento en disco. Por lo que las aplicaciones estaban capacitadas para realizar sólo algunos cálculos y usaban una significativa porción de su poder en presentación y en la lógica de interface de usuario. La memoria limitada también restringía la complejidad de cálculos que se podían ejecutar en esas máquinas. Como resultado las interfaces de los clientes a los servidores eran diseñadas para minimizar el uso de CPU y la utilización de memoria en el cliente y maximizaba el uso del poder de procesamiento del servidor.

Avances significativos en el ***hardware*** del cliente, tanto en capacidad como en poder, han provisto plataformas para el desarrollo de avanzadas aplicaciones, con más memoria, capacidad de disco y poder de procesamiento, el ***hardware*** del cliente puede soportar aplicaciones con sofisticadas interfaces de usuario para poder realizar cálculos complejos.

El ***hardware*** moderno del cliente ha establecido el uso de tecnologías avanzadas de presentación tales como ***GUI's*** así como avanzadas tecnologías de desarrollo incluyendo tecnología orientada a objetos.

Las aplicaciones del cliente pueden usar gran capacidad de memoria para retener grandes volúmenes de datos complejos así como poder ejecutar operaciones complejas con los mismos. Las interfaces del cliente deben

tener la capacidad de mover los datos desde los servidores a la memoria disponible de éste eficientemente.

El desarrollo de herramientas debe soportar interfaces del cliente con las siguientes características de ***hardware***:

- Grandes cantidades de memoria en las PCs actuales y futuras así como estaciones de trabajo
- Gran cantidad de almacenamiento en disco en las PCs actuales y futuras así como en estaciones de trabajo
- Sofisticados programas administradores de memoria, tales como los que manejan la memoria virtual, se han hecho posibles por la combinación de modernos sistemas operativos, gran cantidad de memoria y almacenamiento en disco.
- Incremento del poder de procesamiento en las PCs actuales así como en las estaciones de trabajo.

*Tabla 2.4. Características de **Hardware** del Cliente moderno.*

En suma podemos decir que el cliente es una máquina que usa un Sistema Operativo y ***software*** de aplicación, ***software*** de comunicaciones y librerías.

El ***hardware*** del cliente no solamente debe ser una adecuada plataforma para las aplicaciones del mismo, sino que también debe estar disponible para proveer al usuario de los recursos de cualquier parte de la red, a través de los servicios del servidor. El cliente usualmente cuenta con óptimas capacidades gráficas para soportar de forma fácil las interfaces del usuario. Además de soportar conectividad con los servidores.

Plataformas del Cliente

Las computadoras personales han dejado de considerarse una herramienta aislada dentro de una organización, como nos lo mencionan

Elbert y Martyna en su libro Client-Server Computing, las PC's han ganado adeptos gracias principalmente a la facilidad que representa el aprender a manejar las aplicaciones que operan en las mismas. "Virtualmente cualquier computadora personal puede actuar como un Cliente dentro de la Arquitectura Cliente/Servidor "[4] claro que para esto es necesario adecuar el sistema personal a las necesidades de velocidad de procesamiento, capacidad de almacenamiento y memoria principal. Es en este apartado donde presentamos algunas de las opciones disponibles para implementar las plataformas del Cliente y del Servidor en una arquitectura de este tipo.

Sistemas Operativos para la Arquitectura Cliente/Servidor

En la elección del Sistema Operativo y las interfaces de usuario en una plataforma cliente se requiere tomar en cuenta varios aspectos tales como: satisfacer las necesidades de dar respuesta a usuarios, a las aplicaciones y a la red; en este sentido se puede hacer una clasificación de los sistemas operativos para las plataformas del cliente basada en las interfaces gráficas ofrecidas al usuario, las cuales marcan la pauta entre el pasado y presente de los Sistemas Operativos.

- Los sistemas operativos sin ***GUI's*** que emplean el despliegue de texto llano, ejemplos de estos son MS-DOS y UNIX.

- La primera generación de sistemas operativos con ***GUI's*** para el cliente, la cual típicamente esta representada por Windows 3.1 y sus predecesores.

- Los sistemas operativos de ***GUI's*** orientados a objetos donde el sistema operativo y las interfaces gráficas están integradas como objetos. El pionero en esta etapa fue Apple Macintosh, seguido por IBM y su ***OS/2*** versión 2.X y Microsoft con Windows NT.

[4] Client/Server Computing Architecture, Applications and Distributed Systems Management. Bruce Elbert, Boby Martyna. Editorial Arteach House inc., 1994.

A continuación presentamos una tabla comparativa de los tres tipos de sistemas operativos que para Elbert y Martyna toman los elementos fundamentales requeridos por una plataforma cliente :

SERVICIO	Sin GUI	Con GUI	GUI Orientado a Objetos
Consulta/Respuesta	X	X	X
Transferencia de archivos	X	X	X
Jerarquía de tareas	1	2	X
Interprocesos de comunicación	1	2	X
Protección intertareas	1	2	X
GUIs estándar		X	X
GUIs avanzados			X

1. Con Multitareas Solamente. 2. Opcional. X. Cuenta con el servicio.

Tabla 2.5. Tipos de Sistemas Operativos con elementos para una plataforma Cliente

Entre los sistemas operativos que podemos encontrar están los de Apple Macintosh, Windows NT de Microsoft, ***OS/2*** de IBM y UNIX. Cada uno de estos sistemas ofrece un amplio punto de discusión y análisis, por lo cual sólo mencionaremos los aspectos más importantes a ser tomados en cuenta para la implementación en una Arquitectura Cliente/Servidor de algunos de éstos.

Apple Macintosh

Macintosh trata de involucrarse en el mundo de los sistemas distribuidos, tomando las ventajas nativas de sus sistemas personales como el soporte a ***GUI's***, único en su género, y que además optimiza recursos de memoria pues este soporte se aloja en memoria ***ROM*** a diferencia de las PC's compatibles con IBM que consumen recursos de la memoria ***RAM***. En un ambiente Macintosh es posible no sólo trabajar con aplicaciones orientadas

al diseño, la arquitectura o la publicidad como tradicionalmente se piensa, también es posible trabajar procesadores de texto, comunicaciones y desarrollar aplicaciones en sistemas como MPW (Macintosh Programmers Workshop) utilizando lenguajes como Think C que cuentan con librerías y compiladores apropiados. Mediante sus toolboxes (herramientas nativas del sistema) Macintosh permite que código nativo y previamente generado sea reusable en diversas aplicaciones, proporcionando además interfaces gráficas en tercera dimensión y gran variedad de creación de iconos. Los sistemas que ofrecen estas ventajas ya fueron liberados por Apple en sus versiones 7 y 7.1 las cuales incluyen facilidades para compartir archivos entre clientes, trabajar en multitareas y multimedia (Quick time), también ofrecen la facilidad de realizar interfaces con redes ***LAN*** y ***Ethernet*** a través de su protocolo ***AppleTalk***. Las series quadra de Macintosh (605, 607) tienen cualidades que las hacen tan o más poderosas que los procesadores 486 de las compatibles con IBM, lo cual brinda la posibilidad de utilizarlas como estaciones de trabajo o servidores.

Windows NT de Microsoft

Desde sus primera aparición Windows NT se ha visto rodeado de complicaciones, con limitaciones de funcionalidad, lenta ejecución y excesivo uso de memoria. Sin embargo, pese a situaciones como la anteriormente expuesta, los números reflejan que Windows se ha convertido en el ambiente gráfico más usado en el mundo (más de 10 millones de usuarios para 1993 y más de 5000 aplicaciones que ya operan bajo ambiente Windows), y muchos de estos usos los constituyen Front-Ends de Cliente/Servidor. Windows NT trabaja bajo una interface gráfica idéntica a la de su hermana Windows 3.1, corre aplicaciones a 32 bits, provee de ***GUI's*** y acceso a MS-DOS con un manejo de objetos y comunicaciones con procesadores internos y externos, de ***DLL's*** (Librerías Dinámicas que proveen de objetos ***API*** a los programadores), de DDE (Intercambio Dinámico de Datos de programas mediante un protocolo establecido), de ***OLE*** (Objetos Ligados que permiten pasar de una aplicación a otra sin abandonar la primera), además de dar soporte a diferentes plataformas como Intel 386 y 486, Pentium y DEC Alpha entre otros. Como recursos mínimos Windows NT necesita de un procesador 386 o 486 con 40 y 33 mhz respectivamente y 70 mb en disco duro para cargar el sistema operativo.

En Windows NT se pueden conceder o negar permisos de acceso a diferentes partes del sistema, llevar bitácoras de los usuarios y verificar el uso de los recursos por parte del administrador de la red, permite que múltiples aplicaciones corran concurrentemente. Windows NT carece de aplicaciones nativas de 32 bits por lo cual las de 16 bits disponibles tienen problemas de degradación, sin embargo como contraste NT es poderosamente portable en ambientes donde existen diferentes tipos de clientes como mencionamos anteriormente

UNIX

UNIX es uno de los ***sistemas operativos multiusuario y multitarea*** de tiempo compartido más famosos del mundo y muy versátil en plataformas para clientes o servidores. Sin embargo, una de las características que pueden darle más fortaleza en el futuro es el X Windows System, el cual provee de una efectiva ***GUI*** para ambientes Front-End de redes y que es uno de los puntos débiles de UNIX pues lo hace poco amigable. El estar escrito casi en su totalidad en el lenguaje C ha hechó de UNIX una herramienta de dominio público y muy común. Existe también la posibilidad de construir Arquitecturas Cliente/Servidor con poderosos servidores de SUN o IBM corriendo bajo UNIX y utilizar herramientas 4GL aunadas a las intrínsecas del shell de UNIX y el lenguaje C.

El Sistema X Windows es realmente un protocolo de comunicación y un conjunto de herramientas de interfaces que utiliza ***TCP/IP***.

OS/2 de IBM

OS/2 es un sistema operativo ***multitarea*** que trabaja a 32 bits, desarrollado por IBM y Microsoft, que soporta aplicaciones de Cliente/Servidor. ***OS/2*** surgió sobre la idea de realizar un proceso ***downsizing*** de un sistema operativo de ***mainframe*** para el ambiente de ***microcomputadoras***. IBM describe a ***OS/2*** como "Un todo en un sistema operativo " que introduce las siguientes ventajas:

- Un sistema operativo ***multitarea*** que trabaja a 32 bits y con procesadores arriba de 386

- ***GUI's*** orientados a eventos
- Iconos de los recursos de la red
- Ambiente ***multitarea*** que protege aplicaciones

Las ***GUI's*** orientadas a objetos de ***OS/2*** están contenidas dentro del Manejador de Presentaciones (PM) el cual provee de interfaces gráficas de programación que derivan de los sistemas de ***mainframes***. Este manejador trabaja por eventos (clikcs al ***mouse***, entradas de teclado etc.) y se basa en transmisión de mensajes que se asocian con ventanas; existen herramientas 3GL y 4GL que soportan este ambiente y lo cual hacen que sea accesible al mercado. Para emplear a ***OS/2*** como servidor IBM introdujo su programa LAN Server, el cual permite la comunicación con clientes diversos, ya sean ***OS/2***, Macintosh ó MS-DOS y compite en el mercado con productos como LAN Manager y NetWare. IBM ha logrado incorporar a través de LAN Server una amplia variedad de funciones de red tales como: acceso remoto a los recursos y aplicaciones de red , manejo y administración de red, control sobre accesos y seguridad, protocolo de comunicaciones y soporte a programación de APIs, además de brindar la posibilidad de que programas como NetWare corran sobre ***OS/2***. La principal ventaja que encontramos en las características antes mencionadas, es sin duda la posibilidad de llevar a cabo un adecuado downsizing de mainframes sobre plataformas de PC's en una Arquitectura Cliente/Servidor mediante un protocolo de stacks (que proveen conectividad con ***Ethernet LAN***, ***NetBios***, ***TCP/IP***, ***X.25*** y ***SNA/SDLC***, en este caso ***OS/2*** funge como ***gateway*** con un ***mainframe***) y comunicación de ***APIs***.

Algunos sistemas operativos -Aristof's LANtastic por ejemplo- están diseñados para trabajar en pequeños grupos, los cuales en ocasiones consisten solo de algunas PC's conectadas en red; sin embargo los sistemas operativos más comúnmente usados en una Arquitectura Cliente/Servidor son Novell Net Ware, Microsoft LAN Manager, IBM LAN Server y Banyan VInes. Estos sistemas operativos tienen la capacidad de atender a un mayor número de usuarios brindando sofisticados requerimientos operacionales.

Novell Netware

Novell fué una de las primeras compañías en proveer productos para trabajar en red, estos productos aparecieron cuando predominaban sólo los Servidores de Archivos y de Impresión; "Antes de la aparición del concepto Cliente/Servidor, Novell ocupaba el primer lugar en comercialización de productos para red, hoy día Novell ocupa más del 50% de las instalaciones de sistemas operativos para red instaladas mundialmente" [5].Estos antecedentes han provocado que compañías como IBM y DEC provean a sus productos de interfaces para Novell. Novell desde su primera aparición a principios de los 80's estuvo a la vanguardia de los sistemas operativos, entre otras por las siguientes características:

* Fué el primero en soportar múltiples topologías, y conectividad con sistemas IBM; Digital, Apple y UNIX.

* Fué el primero en soportar sistemas operativos como OS/2 y DOS en todas sus versiones.

* Introdujo el Sistema de Tolerancia a Fallas (SFT) y capacidades de disco espejo.

* Novell NetWare no necesita el soporte de otro sistema operativo para el manejo de múltiples tareas (LAN Manager y LAN Server, por ejemplo, requieren de OS/2 y UNIX respectivamente).

* El concepto de NLMs introducido por Novell, el cual permite que diversas aplicaciones -sobre todo servidores de bases de datos- , las cuales tienen diversas clases de API's no soportadas por Novell, puedan corren sobre éste como NLM's, los cuales por funcionar como extensiones del sistema operativo se ejecutan con mayor rapidez.

---

[5] Client Server Computing. Commercial Strategies. Caroline Cheppell, et all; OVUM Ltd; 1991, p176

LAN Manager

LAN Manager y LAN Server son productos desarrollados por Microsoft e IBM respectivamente, pero que conservan características similares principalmente debido a que se desarrollaron en conjunto y los dos trabajan con OS/2.

LAN Manager es usado en Arquitecturas Cliente/Servidor que tienen una base de OS/2 en los servidores y conserva las siguientes características:

* Al trabajar OS/2 en conjunto con LAN Manager permite que el sistema operativo sea multiusuario y no sólo multitareas.
* LAN Manager se adapta sin problemas a cualquier tamaño de red.
* Soporta conexión con Novell NetWare.
* Soporta diversos protocolos como ***TCP/IP***, ***Net Bios*** y ***Named Pipes***.
* Soporta diversas arquitecturas como Token Ring, ***Ethernet***, FDDI y Arcnet.

PROTOCOLOS DE COMUNICACIÓN

Los protocolos usados en la Arquitectura Cliente/Servidor emplean una o varias técnicas para el envío de mensajes. Esta sección hace referencia a protocolos usados en Cliente/Servidor particularmente :

• ***NetBios***
• ***Named Pipes***
• ***DCE***-RPC (que tratamos anteriormente)
• ***APPC*** de IBM
• ***TCP/IP***

Tabla 2.6. Protocolos de la Arquitectura Cliente/Servidor

NetBios (Network Basic Input/Output System)

Es un protocolo comúnmente usado como un Application Program Interface (***API***) para ambientes de redes locales. ***NetBios*** actúa como una interface entre las capas de ***transporte*** y de ***sesión*** del modelo OSI, las cuales son definidas por el protocolo Net BEUI. ***NetBios*** funciona como un protocolo de transmisión full-duplex que soporta la transmisión de datos por circuitos virtuales y servicios conection oriented y conection less oriented. ***NetBios*** corre sobre MS-DOS y ***OS/2*** , es soportado por ***TCP/IP*** y es posible trabajarlo con productos como LAN Manager y Novell Netware. Así mismo ***NetBios*** puede soportar Token Ring, ***Ethernet*** y Arcnet por lo que puede ser considerado como un posible estándar para aplicaciones de redes.

IBM APPC (Application Program-to-Program Communication)

Los ***APPC*** forman parte de un protocolo recomendado por IBM para IPC en la arquitectura ***SNA*** de IBM, que permiten comunicaciones ***punto a punto*** entre cualquier programa en un ambiente de esta arquitectura. Las entidades en este ambiente usan ***APPC*** para realizar funciones de comunicación que son llamadas por programas de transacción (TP). Un programa puede contener uno o más TP's, cada uno de los cuales utiliza un verbo para llevar a cabo una operación. Además de los TP's, cada ***nodo*** de la red de una arquitectura ***SNA*** contiene una Unidad Física (PU) la cual maneja los ***buffers*** de datos y las ligas de la red. Esta unidad física determina las funciones provistas por los ***nodos***, éstas funciones pueden ser de ***ruteo***, conexión o direccionamiento de dispositivos.

Otro elemento de este protocolo es una Unidad Lógica (LU) que provee el acceso de los Tp's a la red, muchas unidades lógicas puede actuar simultáneamente o bien muchos Tp's pueden accesar los servicios de una unidad lógica. Una de las facilidades y ventajas que presenta este protocolo es la posibilidad de que dos ***nodos*** de la misma, cualesquiera que sean pueden comunicarse independientemente de su localización en la red, del sistema operativo o de la plataforma del ***hardware***; de esta forma una estación de trabajo que opere con MS-DOS puede establecer comunicación con un mainframe IBM provistos ambos de ***software*** de soporte.

TCP/IP

Fue desarrollado originalmente en los Estados Unidos de Norteamérica para ser usado en la red Arpanet (ahora conocida como Internet) la cual unía a varias universidades y centros de investigación en ese país, hoy es la red más grande del mundo. Este antecedente, aunado a que no se burocratizó por ninguna organización, hizo posible que ***TCP/IP*** hasta la fecha sea uno de los protocolos de comunicación más aceptados internacionalmente.

TCP/IP es un protocolo que permite :

- La comunicación entre aplicaciones ubicadas en equipos con características diferentes como marca, sistema operativo, etc.
- El envío de mensajes sin necesidad de conexión física directa entre computadoras
- El ***transporte*** de datos de manera confiable debido a que ***TCP/IP*** se encarga de la recuperación automática de errores de transmisión, pérdida de paquetes de datos o fallas en puntos intermedios a lo largo de la trayectoria que sigue la informacion de transmisor a receptor.
- El ***ruteo*** de la información hasta el destino final, sin necesitar una conexión directa entre origen y destino.
- Al apoyarse ***TCP/IP*** de los protocolos de comunicaciones que se utilizan a nivel capa 2 (***Data Link***), permite que se pueda utilizar ***X.25*** o ***Ethernet***.
- Conectividad universal, debido a que cada computadora dentro de una red con ***TCP/IP*** es reconocida universalmente por medio de una dirección que la identifica.

TCP/IP esta formado por los siguientes protocolos:

a) *IP* (Internet Protocol). Es un protocolo no orientado a conexión que brinda el servicio de envio de paquetes a ***TCP*** y otros protocolos. ***IP*** contiene una serie de direcciones de 4 bytes que cumplen con las siguientes funciones:

• Identificar de manera única a cada ***nodo*** de una red o de un grupo de redes.
• Identificar a miembros de la misma red (por tener el mismo prefijo).
• Poder dirigir o direccionar información entre un ***nodo*** y otro, aún cuando esten en distintas redes.
• Poder dirigir o direccionar información a todos los miembros de una red o grupo de redes.

*Tabla 2.7. Funciones de las direcciones de **IP***

De esta forma si un paquete tiene que atravesar diversas redes para llegar a su destino, ***IP*** consulta su tabla de direcciones tanto de redes como de grupos para determinar donde debe mandarse esta información. Sin embargo ***IP*** no cuenta con mecanismos que garanticen la llegada de paquetes o el control del flujo. Estos mecanismos están contemplados en ***TCP***.

b) *TCP* (Transmission Control Protocol). Es un protocolo orientado a conexión que permite y garantiza el envío de paquetes (datos) a los procesos del usuario y regula el flujo de la información. Esto significa que cada mensaje que un ***nodo*** desea enviar, se partirá en paquetes a los cuales se les asignará un número. Para completar la tarea anterior, cada ***nodo*** que recibe uno o varios paquetes, le informa al que está enviando que efectivamente, sí los ha recibido. Toda la información del número de secuencias del paquete, del reconocimiento, del control de flujo y otros datos, esta contenido en el encabezado de ***TCP***.

Estas tareas son posibles gracias a que antes de que se envíe la información entre dos ***nodos***, ***TCP*** establece una conexión, de igual forma cuando se ha terminado de enviar paquetes, se hace el proceso contrario para terminar la conexión satisfactoriamente.

SOCKETS

"Un socket es un objeto abstracto desde el cual los datos son transferidos. Los sockets existen dentro de un dominio de comunicación y cambian datos con otros sockets residiendo en el mismo dominio"[6]. Unix usa el dominio de los sockets para comunicarse entre procesos en un host, considerando el dominio de sockets Internet para soportar la comunicación entre procesos hosts separados.

Los sockets así mismo, son un importante componente de una red con ***TCP/IP*** y parte de la base para construir el sistema de I/O en una red que opere en un ambiente UNIX.

Una red con ***TCP/IP*** puede contener un gran número de sockets, pero cada socket está identificado con una aplicación específica o ***nodo*** específico de la red.

Actualmente las aplicaciones Cliente/Servidor requieren nuevas demandas de los desarrolladores, incluyendo la necesidad del manejo de interproceso de comunicación (IPC). "Este interproceso de comunicación es la transferencia de datos entre programas que están o no localizados en la misma computadora"[7]. Más facilidades de IPC están diseñadas para transferir datos entre objetos a otros procesos.

Ahora trataremos otros componentes de la Arquitectura Cliente/Servidor :

GUI's - API's

Interface gráfica de usuario. Es una de las mayores ventajas que ofrece una Arquitectura Cliente/Servidor ya que brinda la posibilidad de manejar interfaces gráficas entre los usuarios y las aplicaciones, facilitando la comprensión, el manejo y el aprendizaje de las mismas. Otra de las características que hacen de las ***GUIs*** herramientas de trabajo poderosas, es la posibilidad de correr diversas aplicaciones al mismo tiempo y navegar a

---

[6] Working with Sockets. Unix Review.Junio 1994. Vol.12 Núm 6, p45.
[7] Idem p46

través de éstas, abriendo y cerrando ventanas; sin mencionar la facilidad que resulta trasladar archivos con sólo arrastrarlos al directorio deseado.

Las aplicaciones (componente del esquema Cliente/Servidor, del cual hablaremos más adelante) basadas en ***GUI's***, pueden proveer de un gran número de operaciones de una manera más amigable para los usuarios. En una aplicación basada en ***GUI's*** se pueden hacer múltiples diseños de pantallas donde operan menús que se accesan con un dispositivo de puntero (llamado comúnmente ***mouse***), botones de ayuda o de acceso a aplicaciones; con la posibilidad de mantener varias aplicaciones al mismo tiempo y transportar datos y archivos de manera más fácil y cómoda.

La tecnología más conocida en este ambiente es X-Windows, con ésta se pueden desplegar simultáneamente múltiples aplicaciones usando una o más ventanas. X-Windows está basado en la Arquitectura Cliente/Servidor proveyendo de un protocolo de comunicación entre una aplicación y su lógica de presentación. Esto es particularmente exitoso en un ambiente heterogéneo, donde PC's, estaciones de trabajo y otras plataformas pueden correr las mismas aplicaciones.

Las aplicaciones hechas para un ***GUI*** no son portables para otros ambientes de ***GUI***'s. Y para satisfacer está necesidad sería necesario desarrollar un estándar que pudiera operar independientemente del sistema operativo o de la plataforma de ***hardware*** que se tenga y en caso de redes, independientemente del protocolo que se emplee. Sin embargo, hoy en día el estándar de ***GUI***'s es X-Windows, el cual ha servido como modelo para la ***ANSI*** y el Instituto de los estándares ***IEEE***.

Otro concepto importante dentro de las GUIs son los APIs. "Un ***API*** es un método por el cual el desarrollador referencia los servicios que están disponibles en el ambiente y es utilizado para crear la parte frontal de las aplicaciones con respecto a los servicios de la red " [8] (Figura 2.3). Un ***API*** integra un conjunto de funciones y llamadas a funciones que los programadores usan para accesar a los servicios antes mencionados,

[8] Implementing Production-Quality Client/Server Systems.
Bochenski Barbara. Wiley, 1994. p 116.

GUI's pueden ser la creación y ajuste de una ventana, el llamado de una base de datos. Para la construccion de ***APIs*** se recomiendan lenguajes nativos como ***SDK*** de Microsoft, aunque existen otras herramientas con las cuales realizar esta tarea como Presentation Manager de IBM o inclusive en lenguaje C y ensamblador.

Un ***API*** contiene funciones (que realizan tareas como manejo de ventanas, translación de cadenas, creación de sonidos, mensajes etc), estructuras de datos, tipos de datos y archivos. Todos estos elementos permiten que se puedan crear programas que corran en un ambiente windows. Los ***APIs*** están compuestos por tres elementos básicos:

a) La descripción del servicio. Especifica el propósito del servicio y cómo será usado a través de especificaciones de un lenguaje como C, o C++.

b) Tipos de Datos. Son los tipos de datos comunes o definidos por el usuario que se emplearán en las funciones utilizadas.

c) Llamadas a Funciones. Aquí se declaran las funciones que se envían y reciben información.

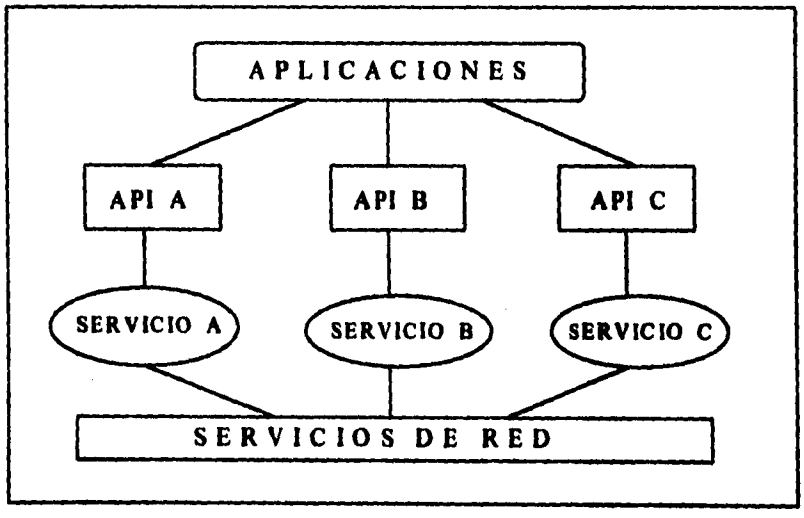


Figura 2.3 Esquema de operación de API's.

LAS APLICACIONES

Para comprender mejor los límites del mainframe así como del servidor de archivos, dividiremos una aplicación en sus partes componentes:

- Lógica de Presentación
- Lógica del Negocio
- Lógica de Datos
- Servicios de Comunicaciones

Tabla 2.8. Partes componentes de una aplicación basada en Cliente/Servidor

Lógica de Presentación

Está provista de un dispositivo que acepta la entrada desde el usuario y despliega cualquier presentación lógica que se llame. Este puede ser un simple dispositivo de caracter con sólo el teclado de entrada o uno más complejo como las imágenes que pueden aceptarla desde un ***mouse***. Controla la interacción entre el usuario y la computadora. Esto es la especificación de ¿qué pasa si el usuario selecciona una opción del menú, presiona un botón, o elige un ***ítem*** de una lista?.

Lógica de Negocio

Es un conjunto de decisiones, cálculos y operaciones que la aplicación debe sacar. Esta puede incluir el cálculo de las compensaciones de empleados, la decisión de aceptar una orden, la evaluación de la aplicación de un préstamo, o los procedimientos tras una transacción tales como una transferencia de fondos.

Lógica de Datos

Es la expresión de las operaciones a ser transformadas en la base de datos que es necesaria para sacar la lógica del negocio. Para las bases de datos relacionales estas expresiones están en sentencias SQL tales como

SELECT, UPDATE, INSERT. Son las acciones que toma el DBMS para extraer la lógica de datos, incluyendo manipulación y/o definición de datos, transacción commit/rollback, etc. Para la manipulación de datos el DBMS típicamente compila en SQL.

Servicios de Comunicaciones

Es la parte que se encarga del envío y recepción de los paquetes de información que viajan a través del protocolo de comunicaciones que se establece al usar un medio físico (cable, ***hub***, ***router***, etc).

SEGURIDAD

La seguridad también es un elemento importante de la Arquitectura Cliente/Servidor, por lo que a continuación hablaremos de los elementos que la conforman.

La seguridad en un mundo distribuido es tarea difícil. La responsabilidad descentralizada puede ser la perdición tanto como un punto único de control puede ser un punto único de falla.

Proceso de Autentificación

Existen tres métodos diferentes por medio de los cuales se puede verificar la autenticidad del usuario: usando algo que conozcan, algo que tengan o algo que sean. Estos métodos se describen detalladamente a continuación.

El método de **"algo que sepan"** está tipificado por el uso del identificador (el nombre de su cuenta) y la contraseña del usuario, o por los sistemas que algunas veces preguntan cierta información personal.

La idea es que este conocimiento no está por escrito, por lo que no es probable que esté al alcance de un intruso. Desafortunadamente, los ID (identificadores) pueden obtenerse fácilmente, y algunas veces las personas conservan sus contraseñas por escrito e incluso las tienen pegadas en sus monitores.

Con el método de **"algo que tienen"** , que añade un segundo nivel de confidencialidad al proceso de Autentificación, puede solicitarle al usuario que tenga en su poder algún objeto físico con el fin de obtener acceso (el equivalente electrónico a una llave). Aunque este objeto puede ser tan simple como una tarjeta de plástico con banda magnética, la solución más común actualmente toma la forma de un generador de contraseñas aleatorias o un dispositivo de reto/respuesta.

Dispositivo de Tipo Ficha: Un tipo de ficha, como son llamados estos dispositivos, proporciona una palabra alfanumérica pseudo-aleatoria que cambia cada minuto más o menos, y que está sincronizada con una base de datos. Esto da como resultado una contraseña que sólo sirve en un momento específico de tiempo y sólo para entrar en sesión una vez.

El nivel final (y el más confiable) de Autentificación, el método de **"algo que sean"**, involucra algún aspecto único e infalsificable de la anatomía de una persona; esta nueva tecnología aparentemente puede leer patrones faciales infrarrojos al paso de las personas usando sólo una simple cámara de video para capturar las imágenes.

Los sistemas biométricos ofrecen el más alto grado de confiabilidad de que el usuario sea realmente quien dice ser, pero también son generalmente los más costosos de implantar.

Kerberos

Kerberos es un sistema basado en encriptamiento diseñado para autentificar usuarios y conexiones de una red. Kerberos tiene a su cargo evitar los accesos no autorizados. Lo hace tan bien, que ahora es casi estándar para efectuar comunicaciones seguras y autentificadas a través de una red.

Sin embargo, no obstante su popularidad, Kerberos no es una solución de seguridad completa, aunque proporciona autentificación de usuarios, no maneja la autorización de aplicaciones ni de transacciones dentro de las aplicaciones. Cualquier determinación de autorizaciones de acceso y de derechos debe ser manejada por otros sistemas de la red.

Una Sola Conexión

Pero en el ambiente de hoy un usuario probablemente necesite tener acceso a varios mainframes, a una red corporativa, a una o más *LAN*, a sistemas especiales de desarrollo, etc. Por esto un solo usuario puede acabar teniendo siete, ocho o más contraseñas. Pero allí es donde fracasa completamente el sistema de contraseñas. Nadie puede recordar 15 contraseñas, todas diferentes, difíciles de adivinar y que no se encuentren en el diccionario. ¿No sería lo ideal que uno se pudiera conectar a un solo sistema, y que por el resto de esa sesión, todos los demás sistemas o redes a los que necesitáramos conectarnos, consultaran una base de datos de seguridad para determinar nuestros derechos, sin necesidad de más procesos de entrada en sesión, interrupciones o contraseñas?. Esto es la unisesión.

El principal problema una vez más es consecuencia del ambiente multiplataforma, multi-sistema-operativo y multiprotocolo, en el cual es sumamente difícil poner en práctica esta unisesión.

Confidencialidad

Aún cuando los controles de acceso sean estrictos y bien mantenidos, se necesita otro mecanismo de control para asegurar que la información de la organización conserve su confidencialidad al ser trasladada a lo largo de los extremos de un sistema distribuido, o hacia otras redes.

Mientras que un sistema envíe datos legibles entre usuarios y servidores, será vulnerable al espionaje sobre la red. La solución a este problema es bien conocida: ***encriptamiento***. Esta sólo transforma la señal para que cualquiera que la intercepte, sin importar cómo lo hagan, simplemente no la pueda leer. El encriptamiento es relativamente barato y puede ser bastante fácil de usar.

Los problemas con el ***encriptamiento*** yacen en el intercambio y en la administración de las llaves del mismo. Por ello se inventó el método de llave pública, con este se da vuelta al problema de la administración de las llaves, usando una llave para el encriptamiento y otra para el ***desencriptamiento***.

Cualquiera que sea el tipo de seguridad que se tenga instalado en los sistemas distribuidos, se tiene que administrar: agregar nuevas personas, eliminar a las que se han ido, modificar derechos y permisos conforme van cambiando los requerimientos del trabajo.

Lo que ha pasado al irse volviendo distribuidos los sistemas y la información es que diferentes departamentos son responsables de diferentes plataformas. La administración en el mainframe es manejada por un grupo diferente del que maneja las *LAN*, y de hecho, quizá sean diferentes personas las que administran cada *LAN*. Y en la mayoría de los casos, la seguridad no es manejada por el departamento de seguridad sino por los departamentos usuarios, donde el administrador de la *LAN* es también el administrador de seguridad.

La responsabilidad por la seguridad de las comunicaciones también está distribuida similarmente. Anteriormente todas las comunicaciones llegaban a una ubicación central, donde un personal bien capacitado sobre la red las monitoreaba. Ahora llegan a cualquier punto de la corporación, sin ninguna administración central.

La seguridad es en gran parte una cuestión de administración, y no es un problema técnico. La primera cosa que se tiene que lograr es convencer de la importancia de contar con un programa de seguridad, con una política general corporativa y con un conjunto de estándares. Obviamente, diferentes sistemas necesitan diferentes niveles de seguridad.

Sistemas Operativos y Seguridad Consistente

Otro factor que influye en la seguridad es la variedad de sistemas operativos a escoger, particularmente de sistemas operativos de red. Algunos tienen más seguridad que otros, pero el problema principal, desde la perspectiva de una red, es la necesidad de dar cabida a múltiples sistemas operativos. Esto genera problemas de compatibilidad y de consistencia.

Es necesario que los sistemas operativos tengan un considerable impacto sobre la seguridad de un sistema distribuido como se muestra en la siguiente tabla:

Concientización	Acceso	Autentificación	Confidencialidad	Administración	Disponibilidad
Establezca una política organizacional con respecto al acceso a las computadoras, a la propiedad de la información y a las responsabilidades del usuario.	Tome precauciones extra si va a permitir el acceso por marcaje (use, por ejemplo, modems de retrollamada y permita el acceso externo sólo a través de un sistema de paredón.	Las contraseñas deben expirar a intervalos regulares.	Encripte los datos transmitidos sobre la red.	Usted debe poder administrar la seguridad (para agregar, eliminar o revisar las autorizaciones de acceso o las listas de control de acceso) desde una estación de trabajo en cualquier parte de la red.	Prepare a sus servidores contra las interrupciones de energía con fuentes ininterrumpibles de poder.
Concientice a los usuarios de su responsabilidad personal de proteger la información de la organización		Use fichas o tarjetas inteligentes como generadores aleatorios de contraseñas.	Encripte los archivos almacenados que contengan datos especialmente delicados o críticos.		Asegúrese de que todos los datos estén respaldados y que sean recuperables en línea, por ejemplo, vía RAID o intercambio inmediato

Tabla 2.9. Elementos de Seguridad

Es importante señalar además que se deben mantener los archivos de respaldo en ubicaciones externas y asegurarse de que los demás sistemas puedan seguir operando si falla algún disco o servidor.

Middleware

Algunos autores manejan este concepto para referirse al ***software*** que accesa a los datos mientras que otros lo usan como el ***software*** que conecta dos módulos, permitiéndoles comunicarse entre sí. De una forma más formal podemos definir al Middleware como un conjunto de servicios que hace posible que los desarrolladores de aplicaciones cuenten con interfaces o ***API's*** que facilitan el desarrollo de aplicaciones. Por ejemplo, para un módulo del cliente que necesita comunicarse con un módulo del servidor algunos autores usan el término "middleware" para referirse al ***software*** que realiza esta función. La parte del middleware esta contemplada en el modelo OSI, en las capas 5, 6 y 7.

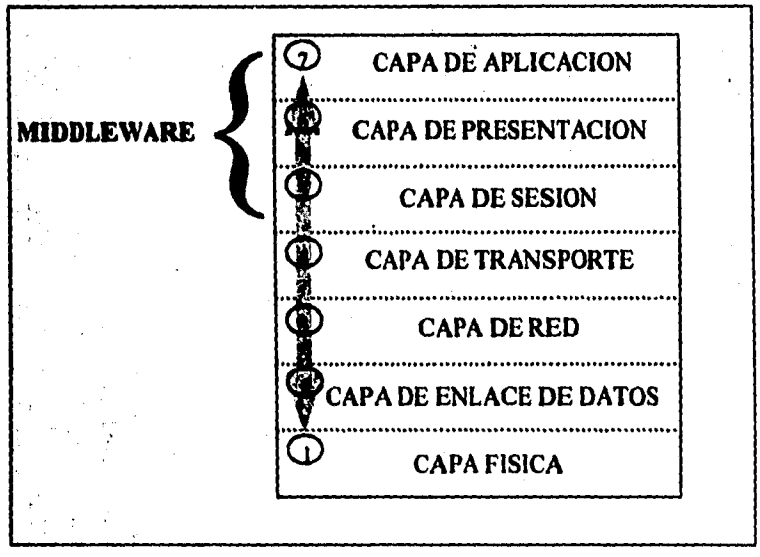


Figura 2.4 Capas donde opera el Middleware en el modelo OSI.

La presencia y utilización del middleware provee de las siguientes ventajas en un sistema distribuido:

- Acceso a servicios de aplicación disponibles en la red. Los Servicios de aplicación son aquellos que proporcionan servicios comunes de información o capacidades de procesamiento para múltiples aplicaciones, tales servicios pueden ser:

  a) Servicios de Mensajes. Los cuales proveen la habilidad para enviar y recibir mensajes entre y a través de las aplicaciones de los usuarios finales.

  b) Servicios de acceso a bases de datos. Los cuales permiten consultar y recibir información de bases de datos locales o remotas.

  c) Servicios de Procesamiento Distribuido. Estos permiten que el procesamiento se distribuya a través de la red por medio de RPCs o estructuras de ***SQL*** .

  d) Servicios de Directorio. Los cuales constituyen un mecanismo para la identificación de usuarios, aplicaciones y sistemas y ayudan a simplificar tareas de direccionamiento.

  e) Servicios de Seguridad. los cuales brindan autenticidad, control de acceso y confidencialidad para las aplicaciones y otros servicios del middleware.

- Las aplicaciones que son desarrolladas con interfaces consistentes de middleware son portables y reusables a plataformas que conserven esas interfaces, además de ayudar a reducir el tiempo de desarrollo de las mismas.

- Propiamente diseñadas e implementadas, el middleware provee acceso a los servicios integrales de la red requeridos por las aplicaciones. La integridad de estos servicios constituye la clave para reducir la complejidad de los sistemas distribuidos.

Para redondear el tema sobre la Arquitectura Cliente/Servidor a continuación se muestra una tabla comparativa entre el modelo tradicional y dicha arquitectura.

SISTEMA TRADICIONAL	CLIENTE/SERVIDOR	VENTAJAS DE CLIENTE / SERVIDOR
Centralizado	Distribuido	Inteligencia y proceso distribuido. Disponibilidad rápida de información crítica. Competitividad
Sistema cerrado	Sistema abierto	Portabilidad de datos, aplicaciones. Libertad del comprador
Mainframe,Minis, Host-terminales	Pc´s, *LANS*, Interfaces gráficas, estaciones de trabajo y servidores RISC/Unix,RDBMS	Precio/rendimiento. Mejor valor. Costo
Base de datos jerárquico	Base de datos relacional y distribuida	Flexibilidad y facilidad de acceso a información
Interface de base de datos propietaria	Estándares empresariales para bases de datos relacionales	Acceso universal
Lenguajes de 3a. generación	Gráficas/Orientadas a objetos, lenguajes de 4a. generación y herramientas *CASE*	Configurabilidad y modularidad de aplicaciones. Rapidez de desarrollo y facilidad de modificación de aplicaciones.
Interface para el usuario de caracteres	Interface gráfica de usuario	Facilidad de uso y capacitación. Productividad
Soporte de sistema operativo único	Estándares de interfaces para programación de aplicaciones (APIs)	Portabilidad de sistemas abiertos

Tabla 2.10. Ventajas de Cliente/Servidor sobre Sistemas Tradicionales

Una vez analizados los elementos de Cliente/Servidor, en el siguiente capítulo describiremos las arquitecturas de redes sobre las cuales se puede construir un modelo de este tipo, así como de la Arquitectura Cliente/Servidor desde dos perspectivas.

3

CLIENTE/SERVIDOR EN DOS Y TRES PARTES

- INTERCONEXION DE SISTEMAS ABIERTOS (OSI/ISO)
- PROTOCOLO DE INTERNET (TCP/IP)
- ARQUITECTURA DE REDES (IBM/SNA)
- ARQUITECTURA CLIENTE/SERVIDOR EN DOS PARTES
- ARQUITECTURA CLIENTE/SERVIDOR EN TRES PARTES

Uno de los elementos fundamentales en un sistema distribuido es la arquitectura de redes elegida, pues esta servirá como el medio de comunicación entre los sistemas y las aplicaciones, por ello antes de tratar las arquitecturas en dos y tres partes de Cliente/Servidor hablaremos de los tipos de redes más adecuados para éstas.

"Una moderna red ofrece un completo conjunto de ***backbones***, mecanismos de transporte, procedimientos y protocolos que constituyen un prerequisito para analizar, diseñar y desarrollar una Arquitectura Cliente/Servidor"[1].

El definir una adecuada arquitectura de redes para una Arquitectura Cliente/Servidor es una de las tareas que requieren más esfuerzos de análisis por parte de los desarrolladores; para esto se deberá tomar en cuenta las características, las facilidades y las estrategias de implementación de una arquitectura de red, tomando en cuenta los antecedentes, el presente y la evolución de dicha arquitectura.

A continuación expondremos una breve descripción de los tres principales tipos de redes las cuales por sus características son capaces de satisfacer los requerimientos de un sistema distribuido y en particular de una Arquitectura Cliente/Servidor en cuanto a mecanismos de transporte.

- Interconexión de Sistemas Abiertos (***OSI/ISO***).
- Protocolo de Internet: Transmission Control Protocol/Internet Protocol (***TCP/IP***).
- Arquitectura de Redes (IBM/***SNA***).

[1] **Client /Server Computing, Bruce Elbert - Bobby Martyna, Artech House, p 39**

INTERCONEXION DE SISTEMAS ABIERTOS (OSI/ISO)

El modelo de referencia de Interconexión de Sistemas Abiertos ***OSI***, constituye una formal descripción de una arquitectura para la comunicación de sistemas abiertos. Existen 7 capas de protocolos en el modelo ***OSI*** (ver figura 3.1) las cuales permiten que hayan diferentes niveles de abstracción según el tipo de funciones a realizar, estas capas fueron especialmente definidas para minimizar y optimizar el flujo de información a través de las interfaces sin afectar los servicios de red de cada capa. "El modelo ***OSI*** (en su presentación de ***OSI*** 7498) es usado como una estructura para la implementación de redes de manera general y que ha logrado ser comúnmente aceptada"[2]. Esto se debe principalmente porque la estratificación de capas y los servicios brindados por ***OSI*** están presentes en la mayoría de las arquitecturas de redes disponibles a nivel internacional.

La estructura general de ***OSI*** se encuentra dividida en dos bloques de capas:

- Las capas inferiores y
- Las capas superiores

El primer bloque lo conforman las capas ***física***, de ***datos***, de ***red*** y de ***transporte***, las cuales tienen como principales funciones establecer la comunicación de la red, la depuración y el ruteo de datos con servicios conecction oriented y conection less oriented, es decir tratar los aspectos a nivel físico principalmente; mientras que el segundo bloque esta compuesto por las capas de ***sesión***, de ***presentación*** y de ***aplicación***, y cuyas principales funciones son el brindar servicios orientados al usuario final (transferencia de archivos, terminales virtuales, correo electrónico etc.), proveer la sintaxis y semántica adecuada para algunos sistemas en particular, coordinando diversas arquitecturas o sistemas operativos, y manejar el intercambio de datos entre sistemas abiertos estableciendo servicios de sincronía entre las capas superiores e inferiores.

[2] Idem p 45

Bajo este esquema se presentan dos conceptos importantes que son los "Sistemas finales" y Los "Sistemas Intermedios"[3]. Los primeros son los que directamente soportan y guardan las aplicaciones, por lo cual se les conoce como ***Hosts*** porque éstos son huéspedes de las aplicaciones de los usuarios finales, las capas inferiores algunas veces se encuentran en los sistemas finales pero solamente actuando como soporte a los mismos, sin contener funciones de los sistemas intermedios. Por otra parte los sistemas finales sólo incluyen las capas ***física***, de ***datos*** y de ***red*** realizando funciones de ruteo, almacenamiento y funciones de intermediación de redes por medio de ***stacks***.

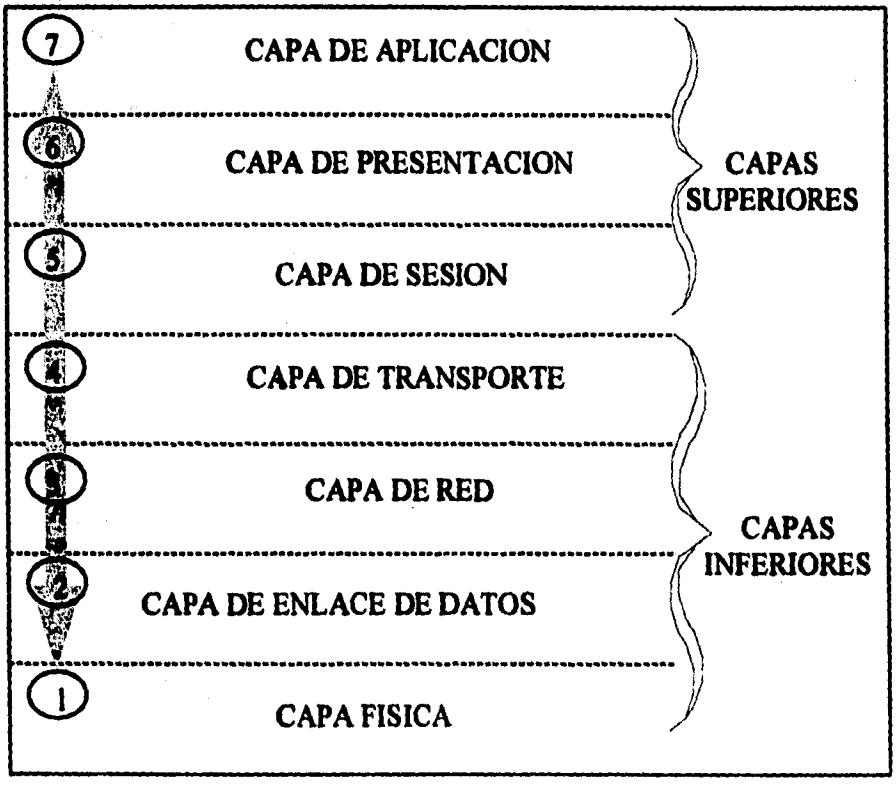


Figura 3.1. Capas del Modelo OSI

[3] Idem p 49

PROTOCOLO DE INTERNET (TCP/IP)

El conjunto de protocolos conocido como ***TCP/IP*** es muy importante al hablar de arquitectura de redes por su amplia cobertura y lo accesible que resulta contar con su serie de protocolos para brindar servicios de correo electrónico, transferencia de archivos y manejo de aplicaciones en red; estos servicios proveen de un robusto y abierto mecanismo de transportación a través de sus protocolos. Son 3 las principales características que convierten a ***TCP/IP*** en un clásico huésped de aplicaciones distribuidas:

Apertura. Por que las especificaciones de ***TCP/IP*** son y han sido siempre del dominio público, lo cual lo ha liberado de algún vendedor en particular teniendo la posibilidad de soportar diversas arquitecturas de red.
Accesibilidad. Las implementaciones de ***TCP/IP*** son accesibles y a un bajo costo, la disponibilidad que hay de contar con sus especificaciones y código fuente lo ha convertido en el estándar de muchas universidades y familiarizado con sistemas operativos populares como UNIX.
Simplicidad. Al encontrarse los procesos casi estandarizados, muchos de los problemas que pueden existir, se tienen resueltos de antemano, teniendo las soluciones listas y a la mano.

Tabla 3.1. Características de TCP/IP

TCP/IP no esta casado con UNIX como se cree, a pesar de que fue proliferado por éste principalmente. Aunque en sus siglas mismas ***TCP/IP*** se identifican solamente protocolos de transporte y red, el stack incluye aplicaciones que tiene referencia con todos los protocolos de definidos por la ***IAB*** Internet Activities Board (ver figura 3.2). Con ***TCP/IP*** y bajo un esquema Cliente/Servidor se pueden proveer servicios directamente a los usuarios finales o bien servicios de soporte a las aplicaciones. Las

aplicaciones del Cliente tienen conexión directa con las aplicaciones del Servidor vía ***TCP/IP***.

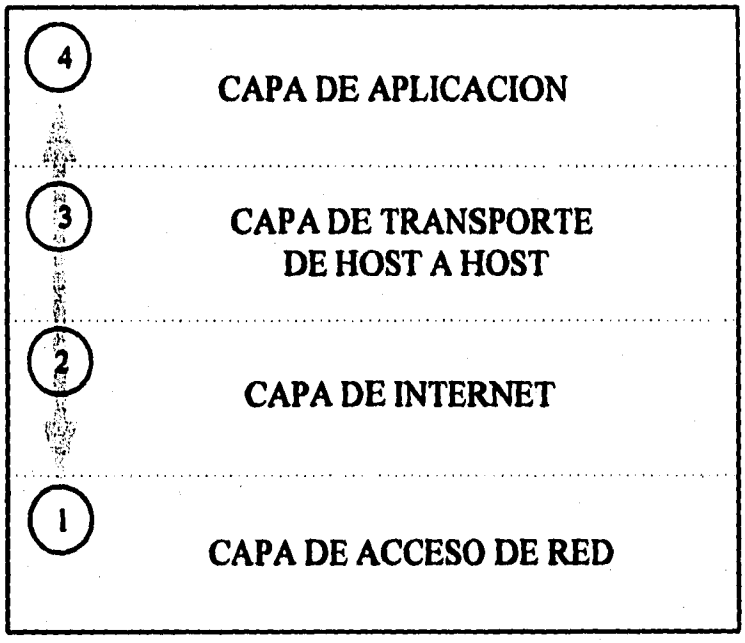


Figura 3.2. Capas de TCP/IP.

TCP/IP contiene en su nivel de aplicación un Protocolo de Transferencia de Archivos (***FTP***), el cual provee de un mecanismo simple para transferir archivos a través de los sistemas de la red en un ambiente orientado al usuario que le permite copiar, borrar, listar y renombrar archivos que pueden ser manejados en ***ASCII*** o binario, ***FTP*** opera en un Arquitectura Cliente/Servidor en el cual un Cliente requiere de ciertos archivos para su procesamiento y ***TCP/IP*** se encarga de coordinar al Servidor y al Cliente para la transferencia a través de un conjunto pequeño de comandos y por medio de lo que se ha denominado un Protocolo Trivial de Transferencia de Archivos (***TFTP***) y que puede alojarse en ***ROM*** por su poco espacio que ocupa en memoria, además de que ***TCP/IP*** en este aspecto provee de sistemas de seguridad parcial o total.

Sun Microsystems desarrollo ***NFS*** (Network File System), que ha pasado al dominio público usando el protocolo ***TCP/IP***, este sistema provee a los usuarios de un servicio transparente de distribución de archivos a través de uno o varios servidores; para tal función ***NFS*** intercepta los requerimientos de archivos en el sistema operativo y a través de ***RPC***s provee el servicio requerido, (Ver figura 3.3).

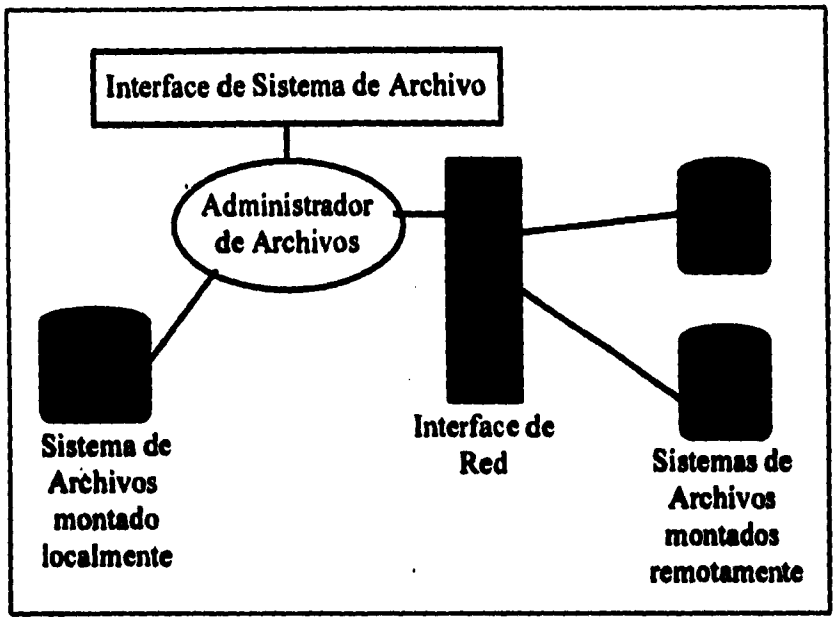


Figura 3.3 . Sistemas de Archivos Distribuidos

Uno más de los servicios de aplicación proveídos por ***TCP/IP*** es a través del Protocolo de Terminal Virtual (figura 3.4) , el cual permite a un usuario conectarse a un sistema remoto y emular una terminal virtual (***TELNET***) local a ese sistema, lo cual para un ambiente distribuido reviste vital importancia pues permite compartir aplicaciones y programas estableciendo los mecanismos adecuados de seguridad a través de privilegios establecidos.

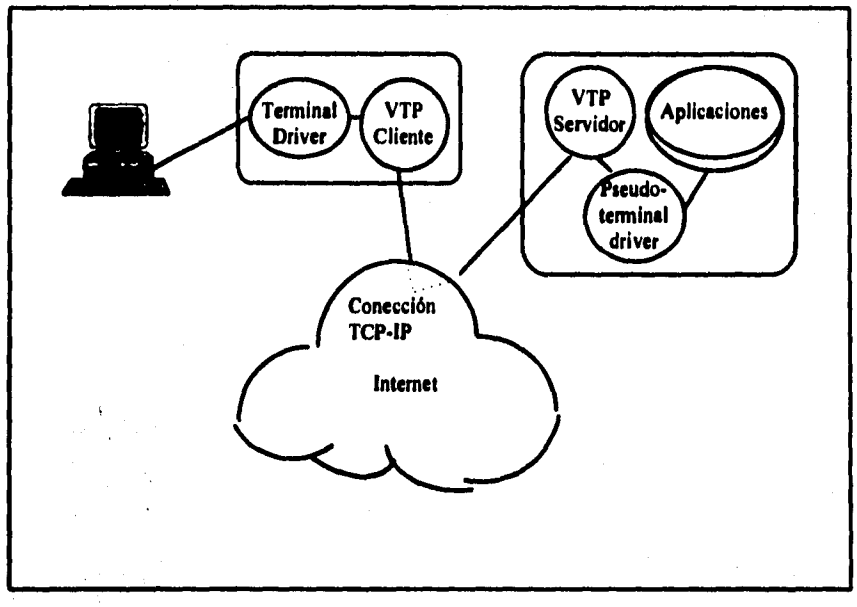


Figura 3.4 . Facilidad de acceso remoto, uso de Telnet.

En resumen podemos decir que las aplicaciones y los protocolos de ***TCP/IP*** son adaptables a un ambiente distribuido por su facilidad y accesibilidad principalmente, convirtiendose en una sólida opción para la implementación de sistemas en ambiente distribuido.

ARQUITECTURA DE SISTEMAS DE RED (SNA) DE IBM

La Arquitectura de Sistemas en Red (***SNA***) fue definida por IBM en la década de los 70's y fué en aquel entonces la primera definición completa de lo que es una arquitectura de red, estableciendo estratos de protocolos que influyeron en el diseño de otras arquitecturas como el modelo ***OSI*** (ver tabla 3.2).

Modelo OSI	SNA
Capa de Aplicación	Servicios de Aplicación -Transacción
Capa de Presentación	Servicios de Presentación
Capa de Sesión	Control de Flujo de Datos
Capa de Transporte	Control de Transmisión
Capa de Red	Control de Ruta
Capa de Enlace de Datos	Control de Enlace de Datos
Capa Física	Control Físico

Tabla 3.2. Comparación SNA Vs. OSI

En las definiciones de ***SNA*** las redes están compuestas por ***nodos*** y ligas que proveen los conductos entre los primeros; el Protocolo ***SDLC*** Control Síncrono de Liga de Datos controla la estructura de transferencia de datos y comunicación entre ***nodos*** adyacentes, este número de ***nodos*** está dado en función del nivel de capacidades de la arquitectura definida. ***SNA*** tiene elementos lógicos llamados Unidades de Direccionamiento de Red (***NAU***'s) que permiten establecer la comunicación entre ***nodos*** vía sesiones, lo cual en el modelo ***OSI*** es realmente una conexión de transporte. ***SNA*** especifica tres tipos de ***NAU***'s: Unidades Lógicas las cuales representan a los usuarios y a las aplicaciones; Unidades Físicas las cuales controlan las ligas físicas y los recursos entre ***nodos***; y los Puntos de Control que monitorean y coordinan a las anteriores y que residen en el ***host*** y a los cuales tiene acceso el administrador de la red.

"Actualmente IBM ha considerado fuertemente la posibilidad de extender el alcance de su arquitectura con otras a fin de buscar ventajas corporativas en el mercado internacional"[4].

Este es un comentario acerca de lo que IBM pretende introducir como estrategia corporativa dándole a su arquitectura la posibilidad de soportar

[4] Idem p 62

sistemas distribuidos y múltiples protocolos mediante la introducción de nuevos NAUs que permiten la comunicación entre puntos que no dependen directamente de un ***host*** y dando apertura para el soporte de nuevos protocolos que no sean de ***SNA***.

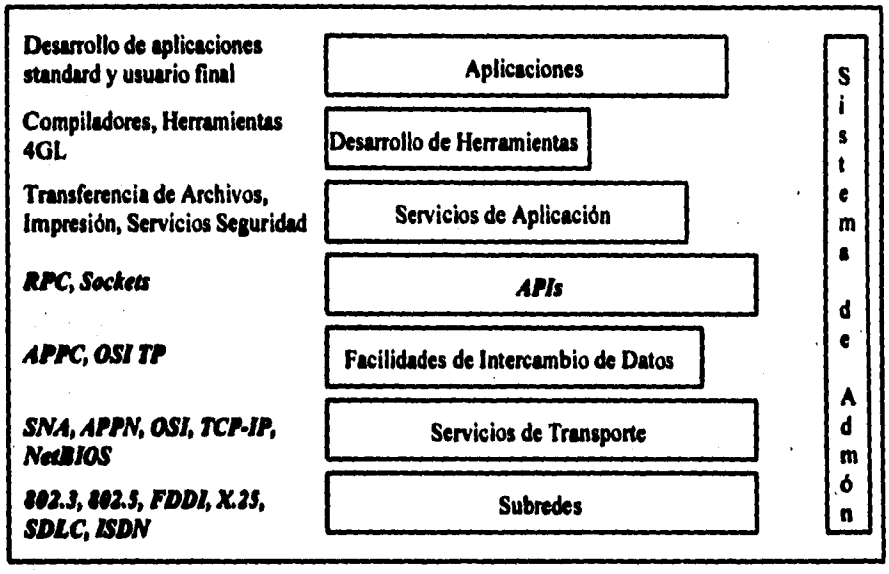


Figura 3.5 . Arquitectura de sistemas distribuidos de IBM

El aspecto de apertura multiprotocolos de IBM a permitido que ***SNA*** trabaje con los estándares tales como UNIX, ***OSI*** y ***TCP/IP*** soportando portabilidad de aplicaciones bajo ***MVS***, VM, OS/400 y ***OS/2***; esta política seguramente será tomada como una tendencia, permitiendo que Cliente/Servidor tenga una amplia gama de posibilidades de diseño e implementación.

Una vez expuestos los tipos de redes que consideramos más importantes hablaremos de los dos enfoques que se le han dado a la Arquitectura Cliente/Servidor.

ARQUITECTURA CLIENTE/SERVIDOR EN DOS PARTES

Generalidades y Ventajas

Muchas configuraciones de Cliente/Servidor actuales usan la arquitectura de dos partes, que consiste de un cliente que requiere servicios desde el servidor. La siguiente figura muestra un ejemplo:

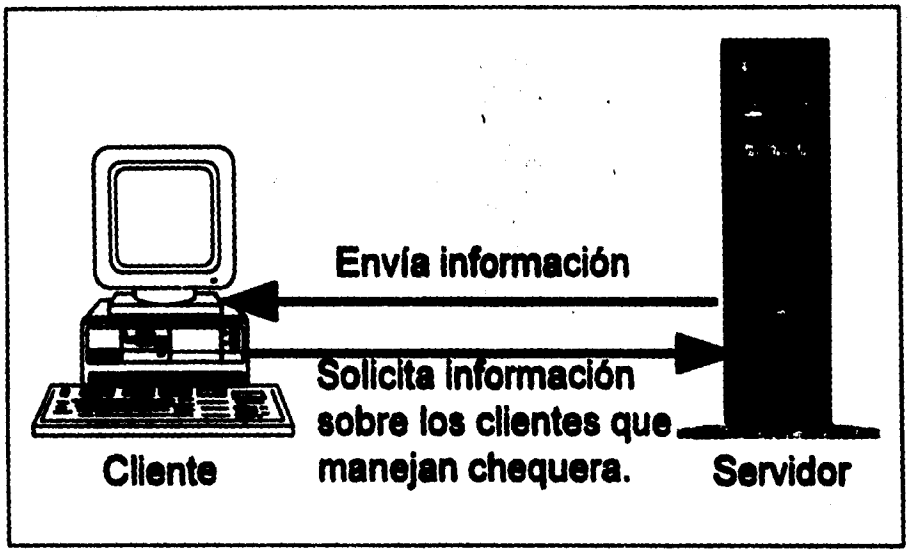


Figura 3.6. Cliente/Servidor en dos partes.

A su vez un Servidor puede funcionar como Cliente de un Servidor diferente, siendo esto todavía un encadenamiento aproximado de dos partes:

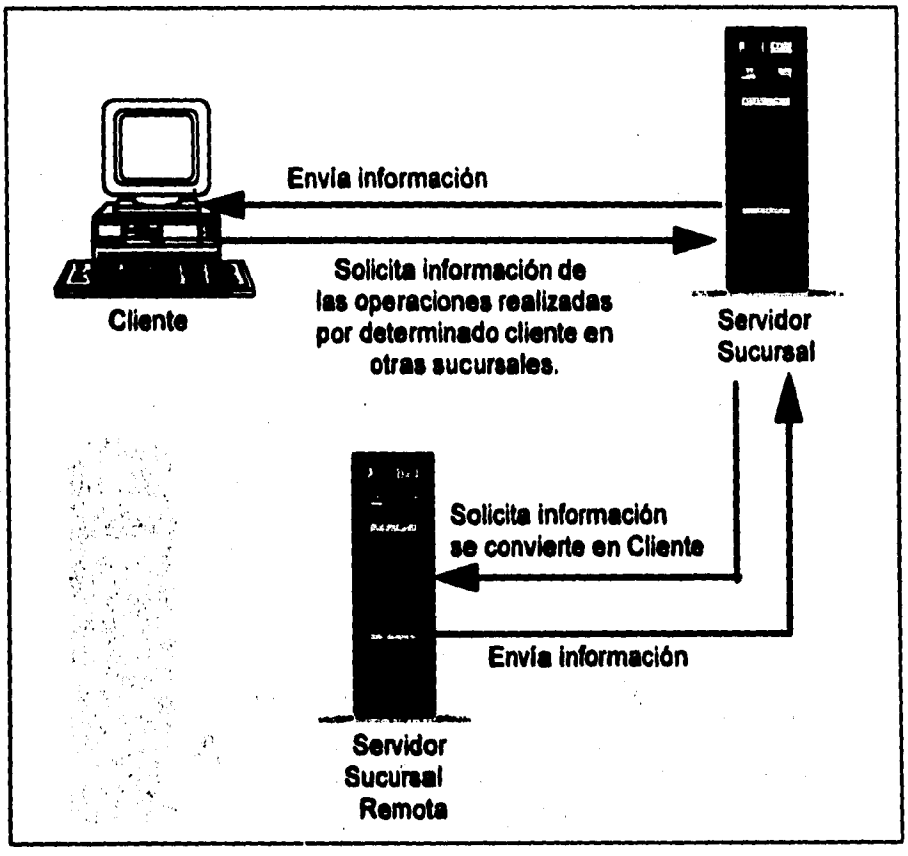


Figura 3.7. Cliente/Servidor en dos partes con Servidor como Cliente de otro Servidor

Como mínimo una Arquitectura Cliente/Servidor asume que los servicios de presentación y la lógica de presentación residen en el Cliente. Esto claramente maneja las inadecuaciones de una interface de terminal permitiendo un manejo local de interface para crear un mejor GUI. La pregunta ahora es dónde va entonces cada cosa?. La primer posibilidad es poner justamente los datos y los servicios de archivo en los servidores y los servicios de presentación, lógica de Presentación, lógica del negocio así como la lógica de datos en el Cliente:

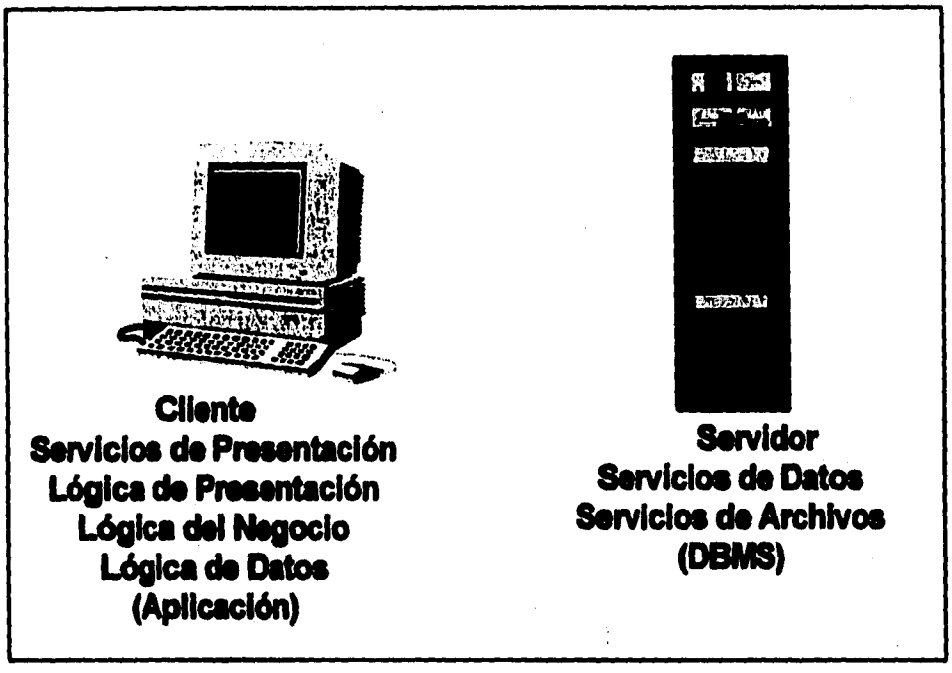


Figura 3.8. Primera alternativa de la colocación de los componentes de una aplicación basada en Cliente/Servidor

Esta aproximación es llamada algunas veces "Sistema de Manejo de Bases de Datos (***DBMS***) remoto". La definición más común de Cliente/Servidor emplea este modelo (la aplicación está en el cliente y el ***DBMS*** en el servidor).

Como se puede observar esto coloca menos demanda sobre el servidor haciendo que el modelo ***DBMS*** remoto proporcione mejor escalabilidad. Sin embargo para aplicaciones complejas donde se involucren lotes de interacción de bases de datos puede poner un fuerte peso tanto en el poder de procesamiento del cliente como en el ***ancho de banda*** de la red. Los resultados completos de sentencias ***SQL*** deben regresar al cliente para el procesamiento pues la lógica de decisión, la cual pudiera haber eliminado algunos de los datos regresados, reside solamente en el cliente. El modelo

de ***DBMS*** remoto también incrementa el peso de la administración de la aplicación pues todo el código de la misma reside en cada cliente, debiéndose cambiar en todos los clientes si existe modificación.

Como optimización a la alternativa anterior se puede reducir la carga tanto en el cliente como en la red moviendo parte de la lógica del negocio al servidor (Figura 3.9.), o bien moviendo toda la lógica del negocio al servidor dejando justamente los servicios de presentación y lógica sobre el cliente (modelo de presentación remota) para minimizar el impacto en el cliente (Figura 3.10.):

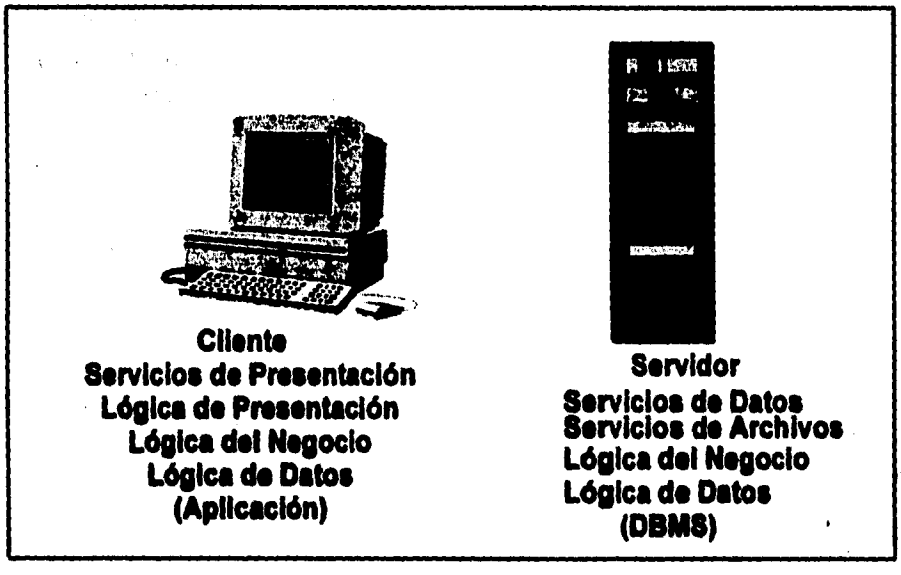


Figura 3.9. Segunda alternativa de la colocación de los componentes de una aplicación basada en Cliente/Servidor

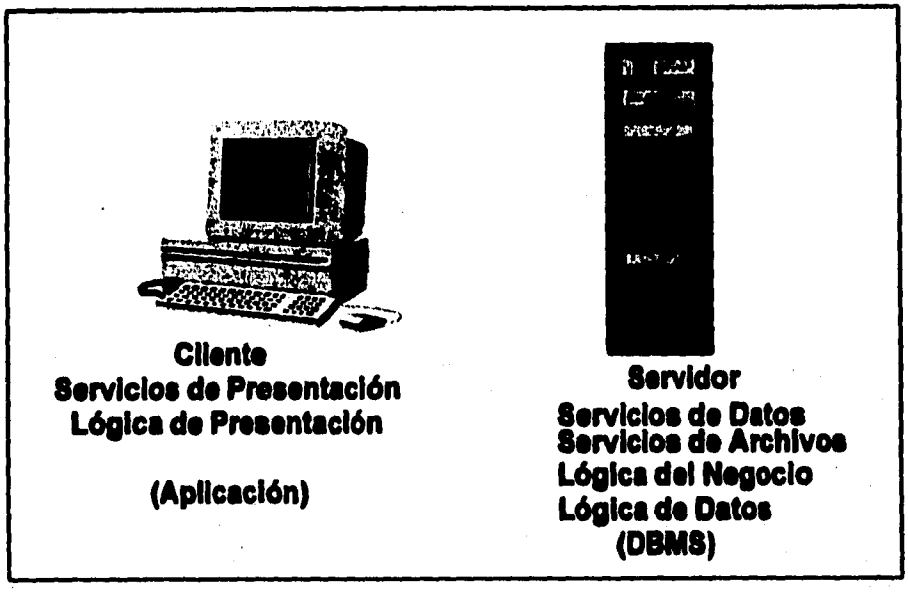


Figura 3.10. Tercera alternativa de la colocación de los componentes de una aplicación basada en Cliente/Servidor

El diseño del sistema debe determinar la localización más apropiada para la Lógica del negocio.

El mecanismo más común usado para colocar la lógica del negocio en los servidores es el de procedimientos almacenados, los cuales son colecciones de código de procedimientos que típicamente incluyen ***SQL*** para accesar la base de datos. El programador explícitamente invoca un procedimiento almacenado por nombre, pasándole los valores de cualquier parámetro que requiera.

Estos procedimientos son compilados en dos partes. Primero el ***SQL*** es procesado, lo cual asegura el análisis del mismo, checando la validación de tablas y nombres de columnas, verificando que el resultado de la consulta

SQL tenga los parámetros requeridos para ejecutarlo, y entonces optimizarlo. La optimización es el proceso por medio del cual el ***DBMS*** usa las definiciones de estructuras de almacenamiento físico (como indexaciones) para determinar la mejor ruta de acceso a los datos.

Segundo, cualquier código de procedimiento debe también ser compilado. El resultado es almacenado como parte de la base de datos. Ejecutando este proceso se ahorra una significante cantidad de tiempo y recursos de computadora.

Pero se debe tener cuidado pues si se sobrecarga de procedimientos almacenados con lógica del negocio altamente compleja se perderán algunas de las ventajas de ejecución. Si no se coloca la lógica del negocio en procedimientos almacenados (como en un modelo ***DBMS*** remoto) pueden usarse simplemente para mejorar la ejecución de la base de datos. En este caso podría almacenarse solamente el compilador de ***SQL*** en el servidor de forma similar al procesamiento de enlace del ***DB2*** (***DBMS*** desarrollado por IBM).

Otro beneficio en el funcionamiento es la carga de red. En vez de enviar las sentencias ***SQL*** desde el cliente hasta el servidor de la base de datos y regresar resultados intermedios al cliente, todo el procesamiento y decisiones pueden ocurrir en el servidor con un simple llamado al procedimiento almacenado. De esta forma el cliente sólo es responsable de ejecutar la aplicación gráfica que ve el usuario, de manejar la interacción con éste y de generar las solicitudes para los datos (en lugar de procesarlos). Mientras el servidor de base de datos se encarga de la administración, integridad de los datos, manejo de transacciones, recuperación de errores, controles de seguridad y del acceso y control de conexiones.

Los procedimientos almacenados también reducen Lectura/Escritura ya que al invocar un procedimiento usualmente se coloca dentro un procedimiento caché así las subsecuentes llamadas no requieren acceso adicional al disco.

Los procedimientos almacenados perfeccionan la base de datos y proveen integridad a la aplicación, ya que los procedimientos son

compartidos asegurando la consistencia de las operaciones y cálculos. Esto también incrementa la productividad pues se escriben y se despliegan los procedimientos comunes en un solo lugar. Los procedimientos almacenados pueden mejorar la seguridad permitiendo al usuario el acceso a la base de datos (DBA) sólo a través de un procedimiento almacenado (no directamente).

Cabe señalar que no es buena idea dejar restringido el dominio del ***hardware*** en la planeación. Un sistema diseñado apropiadamente permitirá que el ahorro en desarrollo, operación y administración equilibre el costo de moverse a servidores o clientes más poderosos (ver Tabla 2.2. Proveedores importantes para Servidores).

Desventajas de la Arquitectura Cliente/Servidor en Dos partes

Las Arquitecturas Cliente/Servidor en dos partes corren con algunos problemas en aplicaciones complejas con muchos clientes, lógica compleja, bases de datos heterogéneas, así como entradas heterogéneas a las mismas.

El primer problema se origina desde la carga de la administración de un gran número de clientes. El lugar de la lógica del negocio en el cliente (junto con los servicios de presentación y lógica) significa que se debe actualizar una aplicación o arreglar un error en cada cliente. Se debe decidir entonces, instalar, y experimentar cada alteración en la aplicación del cliente con sustancial esfuerzo. En particular se debe partir de la falta de uniformidad en las configuraciones del cliente y con la falta de control sobre los cambios subsecuentes a la configuración. Moviendo la lógica del negocio fuera del cliente se minimiza este problema.

El segundo problema está en el intento del uso de procedimientos almacenados para implementar lógica del negocio compleja. Desafortunadamente los lenguajes procedimentales usados en procedimientos almacenados generalmente no están capacitados como lenguajes de programación estándar, no tienen un ambiente de programación con fuertes capacidades en áreas tales como ensayo y seguimiento, control de versión y manejo de librerías. La implementación

para cada procedimiento almacenado debiera usar el lenguaje propietariamente asociado con un ***DBMS*** específico. Sin embargo los procedimientos almacenados no son portables entre ***DBMS***s, restringiendo la habilidad para cambiar ***DBMS***s sin reescribir todas las aplicaciones, también la habilidad de conectar clientes de bases de datos heterogéneas sin escribir procedimientos comunes para cada ***DBMS*** se hace un trabajo pesado.

El uso de procedimientos almacenados para la implementación de la Lógica del negocio también disminuye uno de los principales beneficios de la computación Cliente/Servidor: la Escalabilidad. Como se mueve más lógica de aplicación al servidor se necesita más poder de procesamiento, más rapidez. Cada cliente incrementará el uso del servidor para ejecutar el código de su aplicación. Por lo tanto es cierto que el uso de procedimientos almacenados extienda utilidad y escala del modelo de dos partes mejorando la ejecución, pero la excesiva cantidad de código procedural en el servidor finalmente reducirá el número de usuarios que el sistema pueda ajustar.

El tercer problema que tiene un fuerte impacto es el control de la operación distribuida, dicho control debe ser llevado por un operador (humano), o bien se debe de crear una aplicación que haga dicha función.

Otras desventajas, por ejemplo cómo se pueden correr programas ***batch*** en un sistema de dos partes? el cliente está típicamente paralizado hasta que la tarea del ***batch*** termina, si ésta se ejecuta en el servidor. Además cuando se ejecutan en el servidor la tarea del ***job*** y la interacción de los usuarios muchas veces se interferirán entre sí.

Finalmente, está el problema de las transacciones que se extienden por múltiples bases de datos bajo diferentes ***DBMS***. Cómo puede el usuario asegurar la integridad de una transacción distribuida en una base de datos heterogénea distribuida?. Un ***DBMS*** tiene una habilidad limitada para ejecutar un commit en dos partes (prototipo típicamente usado para asegurar la integridad de la base de datos) con múltiples ***DBMS***s.

ARQUITECTURA CLIENTE/SERVIDOR EN TRES PARTES

Generalidades

La Arquitectura del Cliente/Servidor en tres partes mejora las desventajas de la Arquitectura Cliente/Servidor en dos partes. En este modelo el cliente está dedicado a la lógica de presentación y servicios contando con un ***API*** para invocar la aplicación en media capa (la media capa es un servidor de aplicación sobre el cual se ejecuta la lógica del negocio y desde la cual la lógica de datos invoca los servicios de datos). El servidor de la base de datos está dedicado a los servicios de datos y servicios de archivos (el cual se puede optimizar sin el riesgo del uso de procedimientos almacenados).

La Arquitectura de Cliente/Servidor implica separar las siguientes funciones de aplicación en componentes intercambiables o partes :

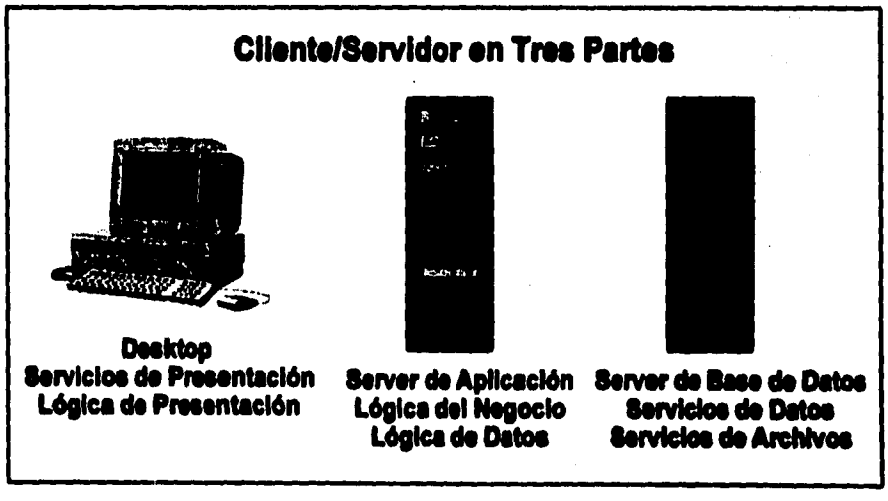


Figura 3.11. Separación de las funciones de una aplicación

- **Presentación** (o interfase del usuario), una fase que soporta la interacción entre humano-computadora utilizando dispositivos tales como el teclado, el ratón y un monitor. También puede tener otras

aplicaciones definidas por el usuario que se comuniquen con la siguiente fase.

- **Funcionalidad, conectividad y servidores de bases de datos,** que corren en una o más computadoras para:

  - Conectar los sistemas ya existentes.
  - Conectarse con nuevas bases de datos.
  - Procesar y formular información.
  - Interactuar con otras interfases de usuario deseadas.
  - Mantener la seguridad,bitácoras y control de versiones

  Los servidores realizan muchas funciones, entre ellas :

  - Validar identidades de usuarios para proporcionar seguridad al sistema.

  - Proporcionar un mecanismo consistente de usuario - transparente para nombrar archivos y directorios en diferentes máquinas y protocolos de sistema de operación.

  - Proporcionar conexiones de comunicación con fuentes externas de información y traducir información a los formatos requeridos por las otras fases.

  - Proporcionar funciones de manejo de información especial

El agregar una conexión de información adicional a la aplicación (como lo sería importar información acerca de oficiales corporativos e instalaciones de una fuente de CD-ROM) es tan simple como conectar un servidor más que pueda comunicarse con información nueva.

- **Datos, sistemas existentes y aplicaciones (Fase de Información)** que han sido encapsulados para tomar las ventajas de su arquitectura con un esfuerzo mínimo de programación.

Esta fase contiene toda la información que requiere su aplicación para poder utilizarla.

Los clientes en la fase de presentación únicamente se comunican con la información a través de los servidores intermedios adecuados. Consecuentemente, los tipos de información que pueden accesar son completamente irrestringidos. Cualquier aplicación puede tener acceso a cualquier tipo de información, de cualquier sistema, en cualquier lugar, siempre y cuando el servidor adecuado esté en su lugar.

Cuando los usuarios interactúan con una nueva aplicación estratégica basada en esta arquitectura, únicamente tienen que enfrentarse con la presentación; los servidores interactúan con las bases de datos o sistemas ya existentes para manipular la información que el usuario requiere.

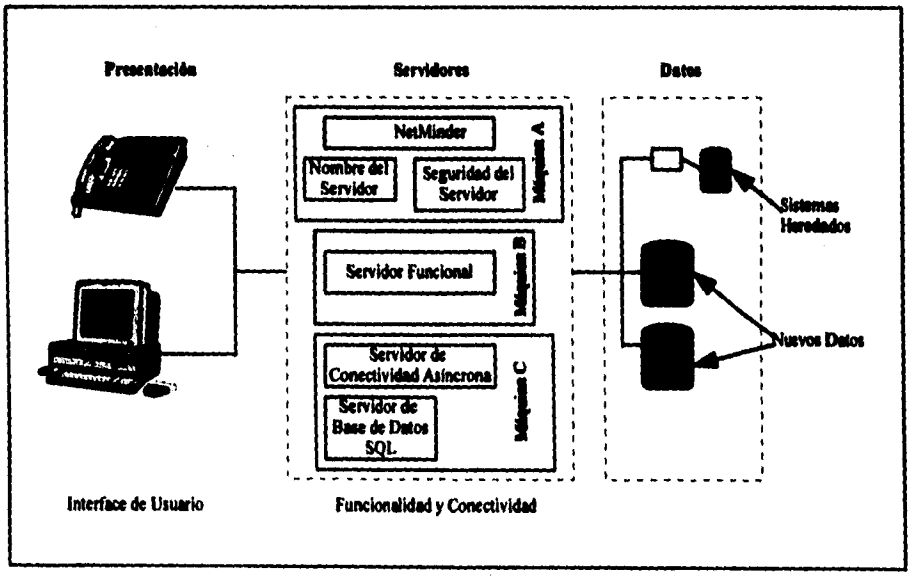


Figura 3.12. Arquitectura Cliente-Servidor en 3 Fases

Pongamos un ejemplo para comprender mejor lo anterior. Supongamos que queremos construir una aplicación que proporcione acceso a información financiera de diferentes fuentes: información del Dow Jones, información del portafolio de acciones en una base de datos SQL, en la compañía de su corredor y la utilización de noticias televisadas de la Red de Noticias por Cable (CNN). Aún más, deseamos tener la capacidad de explotar cada fuente de información utilizando estaciones de trabajo UNIX en nuestros centros de trabajo, nuestras computadoras Macintosh en casa y nuestras PC's Notebook al viajar.

Así es como se apreciaría el sistema bajo el concepto Cliente/Servidor:

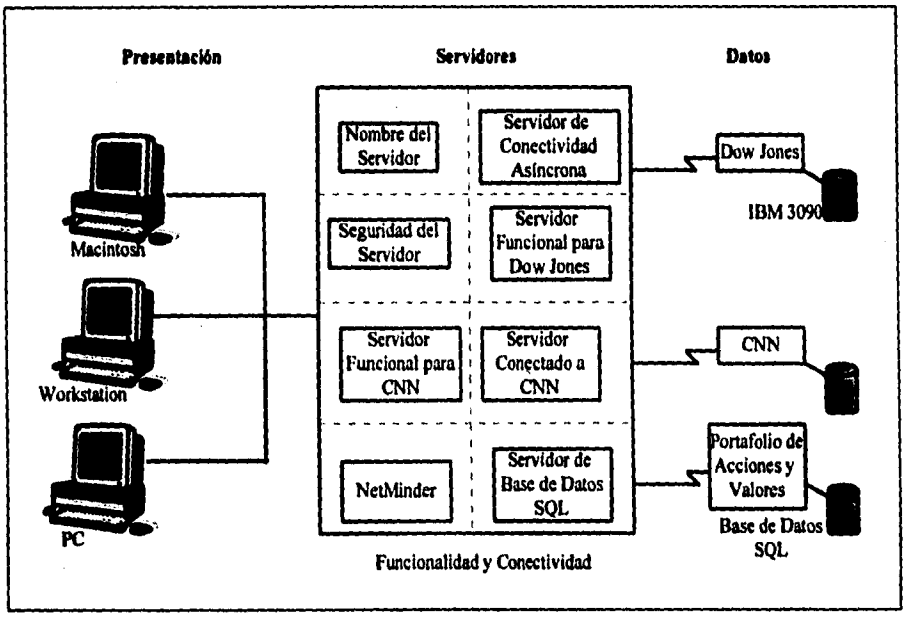


Figura 3.13. Aplicación Financiera

Beneficios de la Arquitectura en Tres Partes de Cliente/Servidor

Al separar una aplicación en tres fases (presentación, servidores de funcionalidad e información), la organización se beneficia con lo siguiente:

- Libertad de seleccionar el sistema de administración de base de datos.
- Respaldo dinámico
- Libertad de seleccionar cualquier interfase de usuario gráfica.
- Flexibilidad para poder añadir nuevas tecnologías conforme se requieran.
- Acceso en-línea a cualquier fuente(s) de información.
- Menor costo del ***hardware***.
- Libertad de seleccionar el ***hardware*** del tamaño apropiado.
- Apertura al valor agregado contínuo.
- Soporte para una estrategia de migración a partir de sistemas antiguos.
- Preservación de su inversión de ***hardware*** y ***software*** actual.
- Desarrollo paralelo de aplicaciones.
- Diferentes enfoques del usuario sobre la información.
- Seguridad.
- Fortaleza y tolerancia a las fallas.

Tabla 3.3. Ventajas de la Arquitectura Cliente / Servidor en tres partes.

Desventajas de la Arquitectura Cliente/Servidor en Tres Partes

Primero la Arquitectura Cliente/Servidor originada por los vendedores de DBMS trataron de proveer toda la funcionalidad necesaria dentro de su propio ***software***. En contraste la Arquitectura Cliente/Servidor en tres partes y la implementación de productos viene desde el mundo de sistemas

abiertos donde la intención fué hacer de Unix un sistema operativo situado en línea para el procesamiento de transacción.

Segundo muchas implementaciones Cliente/Servidor en tres partes no funcionan con el trabajo en grupo y aplicaciones departamentales para lo cual el modelo en dos partes trabaja bien. Como las organizaciones crecen y sus sistemas Cliente/Servidor también, la necesidad de las arquitecturas en tres partes es aparente. Así los productos para estas arquitecturas son relativamente nuevos. En este momento pocos vendedores de ***software*** venden herramientas para construir sistemas en tres partes. Algunos de los vendedores grandes incluyendo Open Environment Corp., provee OEC Toolkit ; Transarc Corp., crea el procesamiento de transacción Encina (TP) monitor y Novell adecúa el sistema Tuxedo como parte de la adquisición de los laboratorios del sistema Unix; Digital Equipment Corp. no sólo soporta Encina y Tuxedo también ofrece su propio ACMS TP monitor.

4

IMPLEMENTACION DE CLIENTE/SERVIDOR

- PASOS PARA EL DESARROLLO DE APLICACIONES BASADOS EN CLIENTE/SERVIDOR
- HERRAMIENTAS PARA DESARROLLAR CLIENTE/SERVIDOR
- IMPLEMENTACION DE UNA ARQUITECTURA CLIENTE/SERVIDOR

PASOS PARA EL DESARROLLO DE APLICACIONES BASADAS EN LA ARQUITECTURA CLIENTE/SERVIDOR

Shaku Atre menciona en su artículo los siguientes pasos para el desarrollo de aplicaciones[1].

1. Identificación de Funciones.

Usando un lenguaje no técnico, liste las funciones a ser incluidas en el nuevo sistema.

La terminología es provechosa cuando se realizan cosas comprensibles, pero dañina cuando es usada para confundir e impresionar a las personas. Una y otra vez se oirá a los usuarios quejarse de que no comprenden de qué están hablando los técnicos. Por lo tanto cuando se identifiquen las funciones a ser cubiertas por el proyecto se debe ser simple y claro.

2. Identificación de Beneficios.

Especificar las metas y los beneficios del proyecto, función por función. Establecer objetivos que definan el incremento de la productividad. Definir cada objetivo, función por función, y ser específico. Las metas funcionales deben ser más tangibles, por ejemplo en el caso de un banco específicamente: las aplicaciones de préstamos deben ser procesadas en dos horas en lugar de dos días).

3. Funcionalidad y Fases correspondientes.

Dividir el proyecto dejándolo en fases y asignar las funciones a ser deliberadas en cada fase.

---

[1] Twelve Steps to Successful Client/Server. Shaku Atre.
DBMS, Vol. 7. No. 5, Mayo, 94, pp 70-72

Desarrollar proyectos Cliente/Servidor es un proceso iterativo y la funcionalidad es deliberada en cada fase. Por lo tanto después de desarrollar una lista de funciones se deberán asignar a diferentes fases del proyecto. En otras palabras la fase uno dará como resultado funciones tales como el procesamiento inicial de una aplicación, comunicar los procesadores de las casas de préstamos, y la habilidad para generar documentos de préstamo automáticamente; la fase dos proveerá otras funciones tales como checar créditos, funciones avanzadas para ayudar a procesar una aplicación de préstamo en línea y acceso remoto al sistema para miembros staff que trabajen en casa, entre otras. Esto pone el escenario para futuras discusiones con los usuarios finales, ya que tal vez prefieran tener la deliberación de ciertas funciones en un orden diferente.

4. Consenso de Objetivos.

Determinar el nuevo objetivo del sistema, administración y objetivos técnicos, identificar similitudes y diferencias entre estos objetivos y buscar un consenso para cada uno de ellos.

Los objetivos no son siempre los mismos para cada grupo. Un objetivo del negocio puede ser incrementar el número de policías asegurados en un 10%, mientras que un objetivo de la administración puede ser deliberar entre los policías con cierto tiempo en la empresa; y un objetivo técnico de ejecución puede ser procesar cada policía dentro de 10 minutos. Se debe tener en mente que los objetivos al trabajar juntos son diferentes, pudiendo haber conflictos entre estos, cabe señalar que los objetivos del negocio siempre son más importantes. La habilidad de encontrar los objetivos del negocio juega el papel más importante en los sistemas Cliente/Servidor.

5. Determinación de Costos.

Desarrollar la mejor estimación de los costos en cada fase implica doblar la cantidad calculada.

Doblar el costo puede parecer excesivo, pero simpre cuesta más de lo que se piensa desarrollar un nuevo sistema. No se debe hacer una propuesta inicial y ponerla como objetivo. Siempre se lleva más tiempo y dinero del que se estima.

6. Determinación del Alcance.

Es sumamente difícil forzar todo para que esté bajo control, es posible que técnicamente no se puedan satisfacer todas las necesidades. Por ello antes de comenzar el desarrollo del sistema es importante determinar hasta donde se quiere llegar y cuales serán las limitantes del proyecto.

7. Ubicación del Procesamiento de Información.

Se debe determinar qué procesos estarán en el servidor y cuáles en el cliente, para ello es necesario desarrollar un modelo de trabajo de lo que hará el sistema.

El modelo del sistema resultante deberá ser el ***Front-End***, pero este sólo es un prototipo. Es importante señalar que este prototipo no se desechará, nada debe ser desechado en los proyectos Cliente/Servidor. Los clientes frecuentemente quieren un prototipo sólo si éste puede ser usado. La gran mayoría de usuarios conocen qué es lo que quieren sólo después de ver el modelo de lo que originalmente pidieron.

8. Decisión de desarrollar o comprar componentes.

Decidir qué Componentes de ***Software*** serán comprados y cuáles se desarrollarán lo cual ayudará para encontrar las herramientas exactas.

Los desarrollos de ***software*** en el ambiente Cliente/Servidor se están incrementando en el último año. Se puede observar ***software*** de aplicación para bancos, finanzas, manufactura, etc. Muchos vendedores actualmente están desarrollando estas aplicaciones. Un número considerable de

vendedores de ***software*** para mainframe están tratando de migrar el ambiente Cliente/Servidor a éstos, para poder seguir compitiendo en el mercado.

Es importante seleccionar y usar herramientas apropiadas. No hay que dejarse maravillar por las herramientas, observe qué sucede detrás de las pantallas de presentación.

Criterios para evaluar el Software Cliente:

a) *Preservación de la autonomía del Servidor.* Permitir al servidor realizar las funciones para las cuales fué desarrollado.

b) *Preservación de la autonomía del Cliente.* El ***software*** cliente debe comportarse y operar de la misma manera cuando esté conectado al servidor de base de datos y cuando se opere con datos locales.

c) *Aplicaciones independientes del Servidor.* El ***software*** cliente debe ser portable y deberá ejecutarse de igual forma cuando accese datos de diferentes servidores de base de datos en diferentes plataformas de ***hardware***, sistemas operativos y esquemas de comunicaciones.

d) *Acceso a las Características específicas de los Servidores.* Las aplicaciones cliente deben permitir vías de acceso a las características específicas de los servidores de base de datos. En algunos casos implementar estas características puede reducir las independencias de las aplicaciones de los servidores.

e) *Impacto mínimo a la estación de trabajo al darle acceso al Servidor.* Accesar los datos del servidor debe impactar mínimamente al uso de los recursos de ***hardware***, memoria y espacio en disco del cliente. El ***software*** cliente no debe requerir

memoria adicional o espacio en disco para procesar o bajar datos del servidor, o bien espacio adicional para replicar el esquema de la base de datos y los catálogos del sistema.

f) *Tener opciones de conectividad lo más completas posibles.* Esta situación debe responderse en términos del número y tipos de servidores de bases de datos por accesar; además de tomar en cuenta la forma en la cual y a que equipos se realizan las conexiones.

g) *Posibilidad de diseñar localmente prototipos.* El ***software*** de desarrollo debe dar la posibilidad de diseñar prototipos y probarlos usando datos y recursos locales, sin requerir de una inversión mayor por encima del servidor de datos, el ***hardware*** de la red y el ***software*** requerido para implementar la aplicación cliente final.

h) *Cubrir en su totalidad el ambiente de desarrollo de aplicaciones.* Para simplificar el proceso de creación de aplicaciones, el ***software*** cliente usado para desarrollar aplicaciones debe tener un conjunto completo de herramientas de desarrollo.

i) *Lenguaje ambiental abierto con posibilidad de modificación.* Es importante que el paquete de ***software*** cliente proporcione robustez y funcionalidad completa para los usuarios de las herramientas; los usuarios también deben tener la posibilidad de cambiar la forma en la cual trabajan un producto en particular. No importa que tan completas sean las herramientas, generalmente no reunen todas las necesidads de los usuarios o también no se acomodan a diferentes niveles de experiencia de los mismos.

j) *Seguir los estándares.* Es importante para el ***software*** cliente el adherirse a los estándares de las herramientas que están inmersas en la arquitectura o plataforma ambiental para proteger la inversión realizada. Ajustandose a dichos estándares

se permite la ***interoperabilidad*** entre varias combinaciones de aplicaciones servidores.

9. Estimación del Impacto.

Definir el impacto del proyecto en la información y en los recursos tecnológicos.

Hay que recordar que se tienen muchos sistemas desarrollados en el ***mainframe***, computadoras personales, etc. Debemos preguntarnos si la misma gente participará en el paso al ambiente Cliente/Servidor, si es necesario contratar personal capacitado o convertiremos a nuestros empleados que manejan cobol, C, pascal, etc., en expertos en técnicas Cliente/Servidor. Las nuevas aplicaciones Cliente/Servidor requieren un cambio de pensamiento. Así mismo, se requerirán de más cambios para guiar estas aplicaciones al ambiente orientado a objetos. Se tiene que incrementar la productividad para manipular aplicaciones complejas quizá de millones de líneas de código. Se necesitarán algunas líneas de programación en ensamblador, para la construcción de librerías. Las herramientas y el ambiente Cliente/Servidor son más efectivos dentro de una programación orientado a objetos y en ambientes de bases de datos.

10. Preparar a la Empresa para el Cambio.

Entendimiento del potencial organizacional y del impacto cultural sobre la organización y desarrollar un plan para el manejo de los cambios. Los cambios en la organización ocurrirán, y es necesario estar preparados para irse ajustando a ellos.

11. Capacitación.

Desarrollar un completo plan de entrenamiento en las nuevas herramientas y aplicaciones.

Un excelente plan de entrenamiento puede incrementar considerablemente los costos, sin embargo los costos por no capacitar pueden ser mucho más elevados. Si los programadores Cliente/Servidor escribirán sus programas desde una ***GUI*** o ***Front-End***, ellos deben aprender a manejar la programación tan bien como la programación orientada a objetos, usando las librerías y códigos existentes. También, el manejo de la red es parte integral de esta Arquitectura, por ello es importante capacitar al personal para el manejo adecuado de la misma.

HERRAMIENTAS PARA DESARROLLAR CLIENTE/SERVIDOR

En este apartado vamos a mostrar y discutir algunas de las herramientas específicas para establecer un esquema Cliente/Servidor. Este conjunto de herramientas de ***software*** es llamado Ambiente Distribuido Abierto (ODE), por la Corporación de Ambiente Abierto (OEC). Y precisamente esta corporación fue fundada para desarrollar la Arquitectura Cliente/Servidor en tres partes de una manera fácil y rápida. En la actualidad las herramientas de la ODE se utilizan en todo el mundo y soportan la mayoría de plataformas y sistemas operativos.

Es conveniente tomar en cuenta el antecedente de que la OSF y UNIX International tienen especificados los estándares sobre Ambientes Distribuidos de Cómputo. Estos estándares incluyen:

- Método para aplicaciones de comunicaciones basadas en RPC's
- Mecanismos de seguridad
- Estándares para ambientes de aplicaciones
- Manejo de un Ambiente Distribuido

Tabla 4.1 . Estándares sobre Ambientes Distribuidos de Cómputo

La OEC tiene desarrolladas herramientas que facilitan el desarrollo de aplicaciones que utilizan estos estándares.

Los grandes comercializadores de ***hardware***, incluyendo IBM, HP, DEC, Sun y Unisys han proclamado su apoyo global para las herramientas ODE y se encuentran interactuando en actividades de mercadotecnia conjunto con la Open Enviroment Corporation. Como resultado, estas herramientas se encuentran ya disponibles con todos los comercializadores, la siguiente tabla presenta algunas de las herramientas disponibles:

Vendedor	Herramienta	Ambiente
APPLE , IBM	Taligent	Mac, Win, *OS/2*
Andersen Consulting Chicago III	Fundación para Procesamiento Cooperativo	*OS/2*
Ask/Ingres Alameda, Calif.	Windows/4GL	DOS, Win *OS/2*, Unix
Bachman Information Systems Burlington, Mass.	Elipse	Win, *OS/2*, Unix
Blue Sky Software Corp. La Jolla, Calif.	Visual SQL	Win, Win 4.x, NT
Bluestone Consulting Inc. Mt. Laurel, N.J.	db-UIM/X	Unix
Blyth Software Foster City, Calif	Omnis 7	Win, Mac
Cognos Inc. Burlington, Mass.	Powerhouse Powerplay Axiant	Dos,Win, Unix, OS/400 VMS, otros Win Win

Tabla 4.2.1. Herramientas para desarollar Cliente/Servidor.

Vendedor	Herramienta	Ambiente
Computer Associates Islandia, N.Y.	Visual Express	Win
Computer Systems Advisers Inc. Woodcliff Lake, N.J.	Silverrun Development Environment	Mac
Dynasty Technologies Inc. Houston, Texas	Dynasty	Win, Mac, *OS/2*, Unix
Easel Corp. Burlington, Mass.	Enfin Enterprise Workbench	Win, *OS/2*, Unix *OS/2*
Forte Software Oakland, Calif.	Forte Development System	
Guidance Technologies Inc. Pittsburg, Pa.	Choreogrepher	Win, *OS/2*
Gupta Corp. Melo Park, Calif.	Gupta SQL Windows 4.1 Gupta Quest	Win, *OS/2* Win, *OS/2*
IBM Corp. Armonk, N. Y.	Visual Age	*OS/2*
Information Builders Inc. New York, N.Y.	Level5 Object 4GL	Win DOS, Win, Unix

Tabla 4.2.2. Herramientas para desarrollar Cliente/Servidor.

Vendedor	Herramienta	Ambiente
Informix Software Inc. Menlo Park, Calif.	Hyperscript	Win, Unix
Inference Corp. El segundo, Calif.	ArtEnterprise Focus	Win, Mac, *OS/2*, NT, Unix, VMS DOS, Win, *OS/2*, Unix.
IntelliCorp Inc. Mountain View, Calif.	Kappa	Unix
Intelligent Environments Tewksbury, Mass.	Application Manager	*OS/2*
Intersolv Rockville, Md.	APS for Client/Server	Win, *OS/2*
Information Resources Inc. (IRI Software Division) Waltham, Mass.	Express ELS	DOS, Win, NT, Unix, VMS, VM, otros
JYACC New York, N.Y.	JAM	DOS, Win, Unix, VMS, otros
Kaseworks Norcross, Ga.	*CASE*:W, *CASE*:PM	Win, *OS/2*
Knowledge Ware Inc. Atlanta, Ga.	Flashpoint Object View	Win, NT Win, NT

Tabla 4.2.3. Herramientas para desarrollar Cliente/Servidor.

Vendedor	Herramienta	Ambiente
Micro Data Base Systems Inc. Lafayetteville, Ind.	MDBS IV	Dos, Win, *OS/2*, NT, Unix
Microsoft	Visual Basic	Dos, Win
Mitem Corp. San Jose, Calif.	Mitemview	Mac
Must Software Intl. Norwalk, Conn.	Nomad	Win, *OS/2*, Unix, VMS, VM, MVS
Next	Nextstep	Unix
Neuron Data Inc. Palo Alto, Calif	C/S Elements	Win, Mac, *OS/2*, NT, Unix, VMS, Open VMS
Open Environment Corp. Cambridge, Mass.	OEC Toolkit 1.1	DOS, Win, Unix, VMS, MPE, Unisys
Oracle Corp. Redwood Shores, Calif.	Oracle Forms Oracle Card	Win, Mac, *OS/2*, Unix, VMS, MVS, VM Win, Mac, *OS/2*
Parc Place Systems Sunnyvale, Calif.	Visual Works	Win, Mac, Unix
Pilot Software Boston, Mass.	Lightship	Win

Tabla 4.2.4. Herramientas para desarrollar Cliente/Servidor.

Vendedor	Herramienta	Ambiente
Powersoft Corp. Burlington, Mass.	Power Builder 3.0 Power Viewer Power Builder Desktop Power Maker	Win Win Win Win
Progress Software Corp. Bedford, Mass.	Progress Version 7	DOS, Win, Unix
Revelation Technologies Cary, N.C.	Seer HPS	*OS/2*
Software AG. Reston, Va.	Natural Engineering Series	Win, *OS/2*, Unix, OS/400, Open VMS, VM, VSE, MVS, otros
Software Artistry Inc. Indianapolis, Ind.	Application Software Expert (ASE)	DOS, *OS/2*, Unix, OS/400
Sybase Inc. Emeryville, Calif.	APT Workbench Gain Momentum	Win, Unix,VMS, otros Unix, X-Term
Symantec Cupertino, Calif	Actor	Win
Texas Instruments Plano, Texas	Information Engineering Facility	Win, *OS/2*, Unix
Trinzic Corp. Palo Alto, Calif.	Forest and Trees	Win

Tabla 4.2.5. Herramientas para desarrollar Cliente/Servidor

Vendedor	Herramienta	Ambiente
Uniface Corp. Alameda, Calif.	Uniface	DOS, Win, Mac, *OS/2*, Unix, VMS, otros
Unify Corp. Sacramento, Calif.	Accell/SQL Unify Vision	Win, Unix, VMS, MVS Win, Unix

Tabla 4.2.6. Herramientas para desarrollar Cliente/Servidor [2]

PLATAFORMAS PARA LA IMPLEMENTACION DE UNA ARQUITECTURA CLIENTE/SERVIDOR

Al implementar una Arquitectura Cliente/Servidor es importante identificar y evaluar las diversas opciones a fin de encontrar la más adecuada para los requerimientos de una organización. Existen dos extremos fundamentales para la creación de una Arquitectura Cliente/Servidor; por una parte las plataformas stand-alone de PC y por otra los ***mainframes***.

Stand-Alone	Mainframes
- Aplicaciones aisladas - Alta potencialidad en procesamiento de textos, hojas de cálculo electrónicas y aplicaciones de bases de datos. - Posibilidad de guardar varias copias de datos en diferentes máquinas - Bajo costo de instalación y de MIPS	- Aplicaciones corporativas - Multiusuario - Mayor posibilidad de manejar los datos pues sólo existe una copia de los mismos - Alto costo en todos sentidos - Control centralizado

Tabla 4.3. Plataformas Stand-Alone y Mainframes

[2] Application Development Trends, Marzo 1994, pp. 31-32

Cada una de estas plataformas ofrece importantes beneficios para una organización, sin embargo también cuentan con limitaciones impuestas por su Arquitectura; por una parte las computadoras personales se topan con el gran obstáculo del almacenamiento, mientras que los mainframes no permiten realizar aplicaciones de manera amigable para el usuario.

Una típica unidad del modelo Cliente/Servidor se presenta en la figura 4.1. En este esquema podemos observar la interacción que tiene el Cliente con el Servidor a través de los servicios que le brinda la red; la forma más común de comunicación entre el Cliente y el Servidor es por medio de un Lenguaje de Consultas Estructurado ***SQL***, el cual permite accesar a un sistema de manejador de bases de datos relacionales distribuidas; al formular la requisición, el cliente que cuenta con recursos propios de procesamiento espera la respuesta del o los servidores que la satisfagan, mientras que la red ya sea ***LAN o WAN*** se encarga de regular el tráfico y de asegurarse que los datos fluyan de manera acorde a las consultas, todo esto permite reducir tiempos de procesamiento en términos reales. Todos estos dependen en gran medida del Sistema Operativo que se integre a la red, ya que éste es el responsable de brindar el acceso a los archivos de la red, el control de entradas y salidas, la conectividad entre los elementos de la misma y el compartimento de recursos.

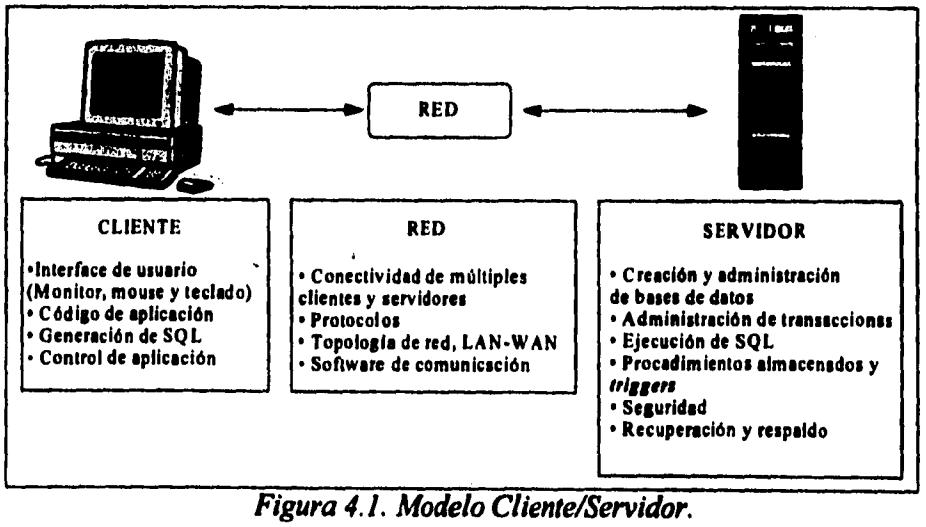


Figura 4.1. Modelo Cliente/Servidor.

Clases de estructuras de Sistemas Operativos que pueden ser usadas en la Arquitectura Cliente/Servidor.

"Existen tres estructuras de Sistema Operativo, la monolítica, la de capas y la de microkernel (o de Cliente/Servidor)"[3], las características de cada una de estas estructuras se presentan en las figuras siguientes.

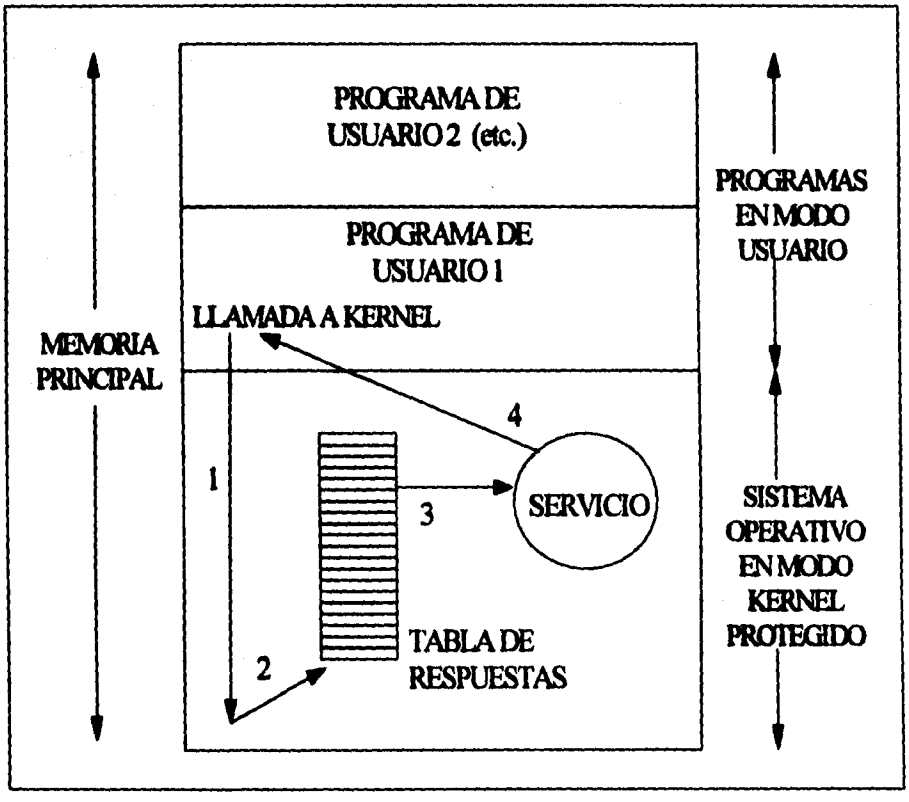


Figura 4.2 . Estructura monolítica de Sistema Operativo

[3] Client/Server Computing, Bruce Elbert ,Bobby Martyna p.112

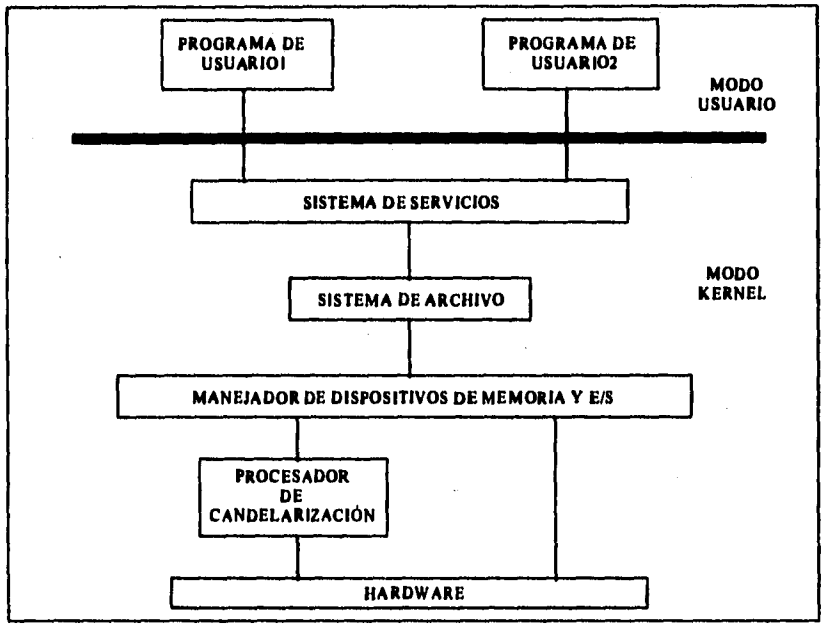


Figura 4.3. Estructura del Sistema Operativo en Capas.

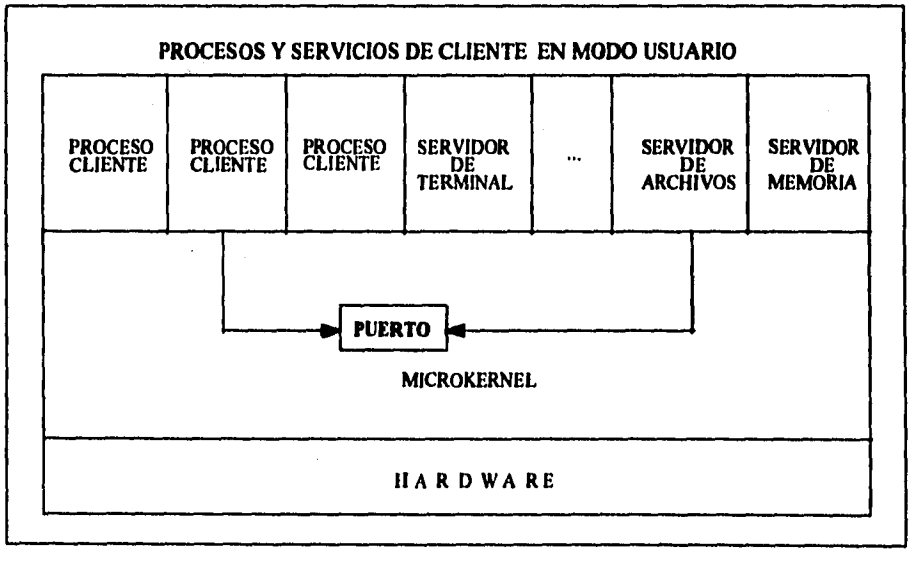


Figura 4.4. Estructura de Microkernel *o Cliente/Servidor*

Podemos observar en la figura 4.1, que la estructura monolítica divide a la memoria en dos grandes segmentos y el sistema operativo reside en el *kernel* el cual se encuentra en modo protegido y los programas de usuario son cargados dentro del espacio de direccionamiento destinado para ello. Esta forma de operación se conoce como de tiempo compartido debido a que el *kernel* tiene la posibilidad de atender en paralelo las diferentes llamadas de servicios que realizan los programas de usuario, todo controlado por un tabla de control de respuestas la cual es accesada por la llamada del *kernel* para dar respuesta a una requisición, esta provee el direccionamiento necesario para satisfacer el servicio y retorna el control al programa de usuario que hizo la petición pero con la alternativa de atender otras en el transcurso de este proceso.

Dos ejemplos de esta forma de operación los encontramos en MS-DOS y en UNIX cuando trabaja en ambiente de PC. Sin embargo el accesar al núcleo por los programas de usuario es un problema, particularmente en sistemas conectados *punto a punto* donde puede haber conflicto al requerir del núcleo similares servicios; este problema es sin embargo resuelto en una estructura de capas (Figura 4.2) donde se provee una protección al direccionamiento múltiple a través de la implementación de jerarquías de servicios de sistema operativo, aunque también en este sentido existe el problema de la congestión de peticiones de servicios cuando existen múltiples clientes con requisiciones y de la degradación del tiempo de respuesta a los mismos.

Es el microkernel sistema operativo de Cliente/Servidor (Figura 4.3) el que pretende resolver ambos problemas, por una parte el adecuar un nivel óptimo de modularidad de servicios y por otra parte ganar en velocidad de procesamiento. Para esto como primera regla debemos proveer a toda la red de interfaces, que permitan manejar los servicios requeridos a través de múltiples núcleos que se distribuyan a lo largo de los procesadores de la red, lo cual facilita el ocupar el espacio de memoria asignado al cliente mismo para cubrir sus requisiciones, Figura 4.4. Aquí el microcanal realiza funciones abstractas de manejo de rutas, procesos, memoria y mensajes, además de brindar la posibilidad de correr paralelamente diversas funciones.

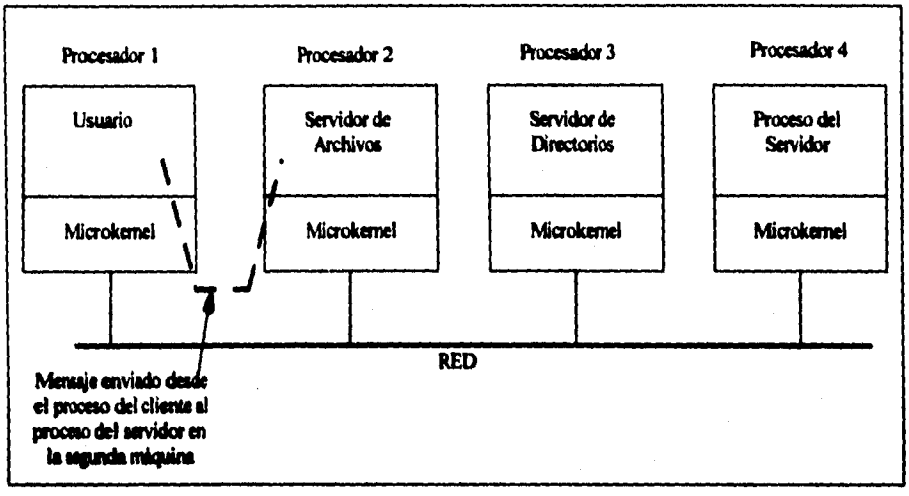


Figura 4.5. El modelo Microkernel *puede ser adaptado a los sistemas distribuidos Cliente/Servidor, donde los procesadores manejan diferentes funciones y los programas de comunicación de la red tienen sus respectivos* Microkernels

Este concepto de microkernel en un sistema operativo es una perspectiva desarrollada por el colegio Carnegie Mellon de la universidad de Pittsburgh Pennsylvania en ambiente UNIX, sin embargo la Fundación de ***Software*** Abierto lo ha tomado como fundamento principal para manejar los requerimientos de varios clientes en forma simultánea de tal forma que compañías importantes desarrolladoras de ***software*** como Microsoft lo ha implementado en el ambiente operativo de Windows NT.

Factores críticos en la implementación de Cliente/Servidor.

Cabe señalar que en la implementación Cliente/Servidor no deben dejar de considerarse los siguientes puntos :

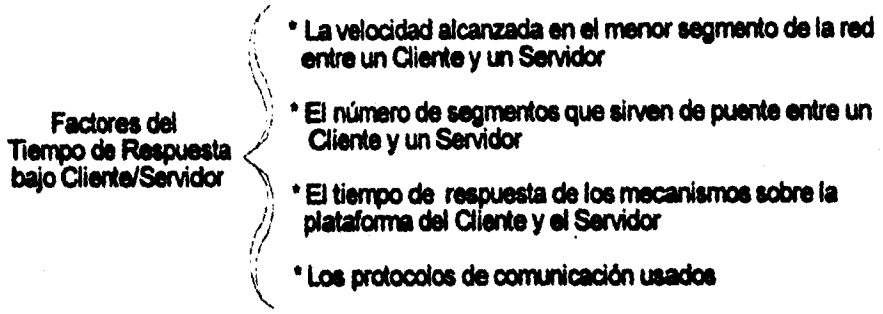
• Tiempo de Respuesta
• Transparencia

Tabla 4.4 . Factores críticos de la Implementación Cliente/Servidor

- ***Tiempo de Respuesta***

En el diseño de un sistema operativo, una gran parte de la atención se centra en el tiempo que el sistema requiere para realizar servicios a los programas de aplicación, lo cual redundará en la eficiencia de la aplicación misma. Estos requerimientos no varían en sistemas operativos para redes, como una regla general usar un sistema operativo de red es solamente práctico si la sobrecarga impuesta a la red es sobre este mismo. Algunos sistemas operativos pueden actualmente ser más rápidos que sus contrapartes no distribuidas. El acceso a los archivos por parte del servidor puede ser más rápido que el acceso a los archivos locales de un cliente, debido por ejemplo a que el servidor puede proveer más buffers de memoria para proporcionar los datos al cliente o permitir al cliente el acceso rápido a disco. Esto ilustra un gran alcance del modelo Cliente/Servidor para los requerimientos de la industria, ya que éste hace posible compartir recursos entre los clientes a través de un servidor y por supuesto se tiene la ventaja de expander los recursos de estos clientes y explotarlos al máximo.

Usualmente el tiempo de respuesta se considera principalmente por la velocidad en un segmento de la red. Pero cuando se ve desde la perspectiva del modelo Cliente/Servidor, éste puede ser considerado de acuerdo a los siguientes factores:



- ***Transparencia***

La Transparencia es una cualidad que protege el diseño de las aplicaciones y los programas, el acceso a bases de datos de forma transparente significa que los programas no necesitan conocer la ubicación física de éstas ya sean locales o remotas, éste proceso se puede llevar a cabo de diferentes maneras, ya sea invocando explicitamente a la base de datos por su nombre o dejando esta tarea al manejador de la base de datos invocando solamente los datos requeridos.

Beneficios de una Arquitectura Cliente/Servidor.

La Arquitectura Cliente/Servidor combina los beneficios de un ***DBMS*** centralizado donde se comparten recursos en un mismo ambiente de minis o ***mainframes*** con las potencialidades amigables encontradas en sistemas personales. Así brindamos altas capacidades de resolución y otorgamos al cliente sus propios recursos para que los servidores se encarguen del manejo fuerte en términos de procesamiento de información.

El cliente concentra aplicaciones y funciones de interface con el usuario. El servidor concentra al ***DBMS*** y lleva a cabo funciones de definición, manipulación, seguridad, respaldo, recuperación y manejo de transacciones.

Ventajas del Cliente:

a) Las PC´s, ofrecen una interface gráfica consistente y amistosa hacia el usuario final.

b) Una excelente relación precio/beneficio en las ***estaciones de trabajo*** tipo PC.

c) Gran variedad de aplicaciones poderosas y fáciles de usar.

d) Los usuarios tienen acceso a información compartida así como a recursos poderosos a través del uso fácil de ***hardware*** y ***software*** de los equipos PC´s.

e) Mientras que el Cliente (PC) usa todo su poder de cómputo, éste no trabaja solo, es apoyado por el servidor o servidores.

Ventajas del Servidor:

a) Los servidores inteligentes comparten la carga de procesamiento con las ***estaciones de trabajo***.

b) Se obtiene un ambiente ***multiusuario***.

c) Una sofisticada administración y seguridad.

Los principales beneficios que aportan los esquemas Cliente/Servidor son el alto grado de rendimiento (performance), la escalabilidad y una plataforma abierta. A continuación expondremos cada uno de éstos:

- ***Performance*** **(rendimiento)**

La Arquitectura Cliente/Servidor está basada en el uso de un servidor de bases de datos, el cual reduce significativamente el tráfico en redes de área local y hace eficiente todo el sistema. Debido en gran parte a que las funciones de las bases de datos y las aplicaciones son llevadas a cabo en diferentes computadoras lo cual permite optimizar el rendimiento de las diferentes tareas.

Por ejemplo un cliente puede ejecutar las mejores interfaces con el usuario sin necesidad de accesar a la base de datos, y en forma inversa el servidor no es responsable de soportar las funciones de interfaces y puede entonces optimizar las actividades propias de un ***DBMS***.

- **Escalabilidad**

Este beneficio aunque no es palpable, a futuro resulta imprescindible desde la perspectiva de que en el modelo Cliente/Servidor es teóricamente posible por ejemplo, reemplazar algún servidor por uno de características más poderosas sin tener que modificar la estructura de los demás elementos o de las aplicaciones que corren en los clientes. Es decir obtenemos una independencia entre el ***Back-End*** y el ***Front-end***, aunque claro tenemos que tomar en cuenta siempre el aspecto de comunicación entre estas partes, esto es, tenemos que preservar las interfaces adecuadas para conservar la integridad del esquema.

- **Plataforma Abierta**

Una vez implementada la Arquitectura Cliente/Servidor, el agregar otro elemento no implica reestructurar el diseño de ésta, ya que la interoperabilidad alcanzada nos brinda este beneficio.

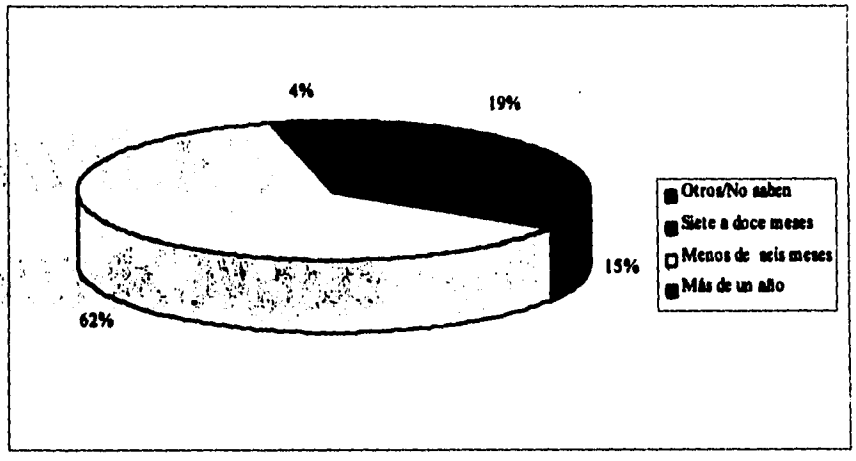
5

CAJERO BANCARIO

- CLIENTE/SERVIDOR EN EL SECTOR BANCARIO
- CAJERO BANCARIO

CLIENTE/SERVIDOR EN EL SECTOR BANCARIO

En la historia la mayoría de los bancos han dependido de un ***mainframe*** de un sistema empresarial, sin embargo, como se muestra en la siguiente gráfica el sector bancario estadounidense está convencido de que uno de los grandes problemas de la información en éste tipo de máquinas es la cantidad de tiempo requerido para contar con el acceso listo y produciendo.



*Gráfica 5.1. Tiempo para tener el acceso listo y produciendo en **Mainframes** en el Sector Bancario* [1]

Este tiempo se considera a partir de los siguientes problemas :

- Carencia de un staff experto
- Dificultad para integrarse
- Consumo de gran cantidad de tiempo
- Dificultad para mantener la integridad de datos

"Desde la década de los 70's, la evolución de la tecnología de bases de datos se fundamentó en integrar y organizar toda la información en una base de datos central y única"[2]. El advenimimiento de las computadoras personales y de los sistemas multiusuarios pequeños, ha hecho que este

[1] Banca Electrónica. High Tech Editores. Número 2 enero de 1994, p.13
[2] Banca Electrónica. High Tech Editores. Número 14, enero de 1995. p. 30.

enfoque cambie. La disponibilidad de estos equipos de herramientas específicas, no disponibles en otras arquitecturas, hace necesario el tener computadores especializados en diversas tareas, esto aunado al dinamismo que hoy día presentan las organizaciones financieras. El contar con una base de datos integrada presenta algunos problemas como:

- El control del tiempo de respuesta de los servicios en línea resulta tecnológicamente difícil.

- Los procedimientos de respaldo frente a desastres requieren tener un computador alternativo con capacidad para toda la base de datos y sus procesos más importantes.

- Impide aprovechar las oportunidades de implantación de *software* adecuado cuando éste no se integra a la base de datos.

Por estas razones, el mercado actual tiende a diversos tipos de máquinas orientadas a aplicaciones específicas que juegan el rol de clientes, captando transacciones para su proceso y las que juegan el rol de servidores, procesando, registrando o ruteando estas transacciones.

Una vez que se acepta el principio de la especialización de los equipos, la herramienta de integración deja de ser la base de datos y ésta es reemplazada por una red de comunicaciones que con fin de garantizar su perseverancia a través del tiempo se basa en tecnologías y protocolos de comunicación ampliamente difundidos.

Los problemas de integración entre aplicaciones son a menudo significativos en las instituciones financieras, y una estrategia de tipo Cliente/Servidor parece en una primera instancia, que aumenta estos problemas. Sin embargo, si se realiza una adecuada asignación de funciones para cada tipo de equipo, es posible mantener e incluso mejorar la integración entre las aplicaciones tradicionales. Esto es posible debido a que tanto los clientes como los servidores pueden lograr que múltiples sistemas preexistentes en el servidor de bases de datos aparezcan integrados al usuario final, sin requerir modificaciones en las aplicaciones.

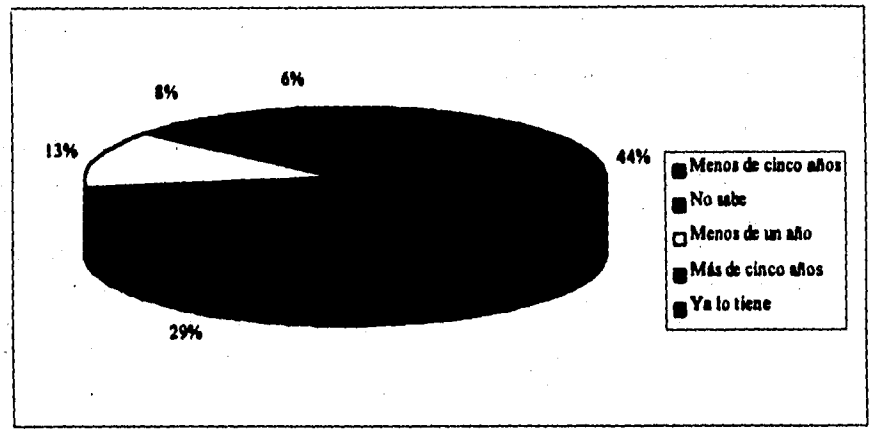
Por otra parte, es indudable que los bancos con su dinámico proceso de automatización siempre van a consumir los recursos de cómputo planeados;

siendo precisamente la Arquitectura Cliente/Servidor quien garantice un crecimiento más fácil y económico para sus necesidades futuras. Así mismo favorece en gran medida la independencia o transparencia de las aplicaciones con los medios de comunicaciones, permitiéndoles incrementar el ***ancho de banda*** o la tecnología de sus redes sin perturbar a las aplicaciones.

La tendencia actual hacia los sistemas Cliente/Servidor favorece en gran medida el uso de las ***microcomputadoras*** con multiprocesadores (ALR Q-SMP, por ejemplo, que tiene 4 procesadores Pentium) y sistemas operativos abiertos; para el sector bancario mexicano estas tendencias tecnológicas ofrecen un panorama alentador y prometedor en cuanto a que proporcionan una solución a los añejos problemas de tiempo de acceso, portabilidad del ***software***, conectividad y tolerancia a fallas.

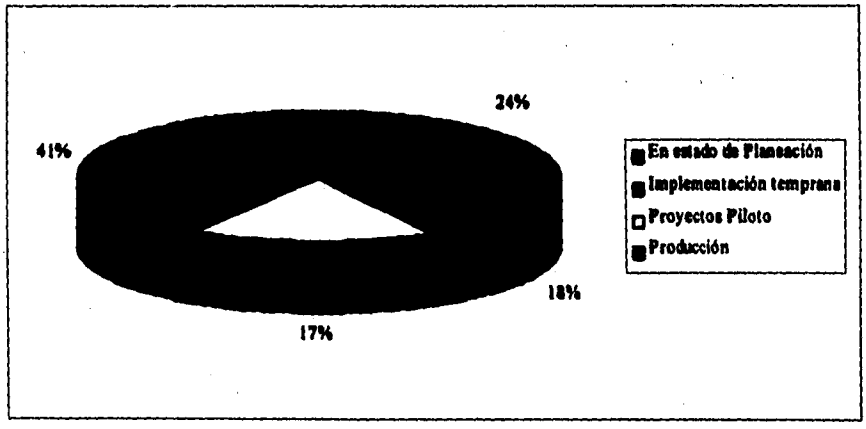
Cliente/Servidor beneficia a las bancos grandes, medianos y pequeños, siendo estos últimos los más favorecidos, ya que pueden contar con sistemas desarrollados en plataformas económicas e ir creciendo conforme sus necesidades lo exijan. Además esta Arquitectura permite liberar el acceso y distribución de información para todo el organigrama de la empresa. Incide por tanto en la mejor automatización de los procesos y en su actualización; utilizando la información que siempre han tenido.

La gráfica 5.2. muestra el tiempo en que las instituciones bancarias estadounidenses planean migrar hacia una Arquitectura Cliente/Servidor, como podemos apreciar más del 50% lo realizará en menos de 6 años, lo cual representa la aceptación de esta Arquitectura por parte de este sector.



Gráfica 5.2. Tiempo en que planean eliminar el viejo sistema a favor de Cliente/Servidor [3]

La siguiente gráfica muestra el ascenso de los sistemas basados en la plataforma Cliente/Servidor. Más del 40% de las empresas ya lo utilizan, mientras que en el resto o está en proyecto o próxima a implementarse.



Gráfica 5.3. Status Cliente/Servidor [4]

En México las instituciones financieras no se han quedado atrás en cuanto a la realización de sistemas basados en una Arquitectura

[3] Idem 1. p.13
[4] Idem 1. p.13

Cliente/Servidor, por lo que a continuación se presentan algunas de éstas instituciones con los sistemas realizados bajo este esquema :

- **Grupo Financiero Abaco :** Decidió reemplazar sistemas que presentaban problemas de coexistencia, uno basado en dos mainframes de IBM que trabajan con ***MVS*** y el otro corriendo con UNIX en minis NCR Y HP, por una Arquitectura Cliente/Servidor con clientes que operan en base a Windows que accesan a una minicomputadora HP que funge como servidor y cuenta con ***SQL Server*** utilizando los siguientes productos: Microsoft Visual Basic, Microsoft Visual C++, Microsoft Windows, Microsoft LAN Manager, Microsoft SQL Bridge, Microsoft ***SQL Server*** y Sybase ***SQL Server***; trabajando con la red denominada AbaNet que se basa en el protocolo de comunicaciones ***X.25***, que está soportada por estaciones satelitales y líneas privadas y diseñada en base a la asesoría de Hewlett Packard de México.

  "Los beneficios obtenidos por el Grupo Financiero Abaco con Cliente/Servidor son los siguientes:

  a) Se eliminó la necesidad de un ***mainframe***, ahorrando unos 200 mil dólares por año.

  b) Se redujo el tiempo de trabajo administrativo de cierre diario en una hora por día.

  c) Se hicieron disponibles desde el mostrador de cualquier cajero, servicios que tradicionalmente eran considerados de alto nivel.

  d) Por tener toda la información de cuentas de ***cheques*** en línea y en tiempo real, se pueden obtener los ***saldos*** de las cuentas al instante.

  e) Contabilización automatizada de transacciones y eliminación de la necesidad de varios niveles de aprobación, lo que permite un servicio más rápido y eficiente." [5]

[5] Banca Electrónica. High Tech Editores. Número 8, julio de 1994. p. 21-22.

- **Bancomer :** Aplicaciones desarrolladas para mercado de dinero.

- **Multibanco Mercantil Probursa :** Sistema de ahorro para el retiro ProgreSAR, Sistema integral de la central de cambios, Sistema central de comunicaciones para banca empresarial.

- **Casa de Bolsa Probursa :** Sistema de asignación automática de mercado de dinero, Sistema de documentación de especificaciones técnicas, Sistema de control de la mesa de mercado de dinero, Sistema de cobertura cambiaria, Sistemas de control de garantías.

- **Bancrecer Grupo Financiero :** Sistema integral de ***crédito***, Sistema de control para el otorgamiento de ***crédito*** con alcance nacional.

  Bancrecer tenía el expediente individual de los clientes para el otorgamiento de ***crédito*** en la hoja de cálculo de Lotus 1-2-3 y era dominio exclusivo del analista. Con Cliente/Servidor se institucionalizó el otorgamiento de ***créditos***. La información pasó a ser dominio del banco y no del analista.

- **Grupo Bursátil Mexicano (GBM) :** módulos de valores, mercado de capitales, contratos, productos y administración.

- **Banco Nacional de México (BANAMEX).** Sistema de Análisis Bursatil, Sistema de Mercado de Dinero, Sistema de Prestación Médica, Sistema de Monitoreo de Cuentas, Modelo de Gestión, Monitoreo, Control y Administración de Redes y Sistema del Centro de Atención Telefónico y operación de Tarjetas.

CAJERO BANCARIO PROPUESTO

A continuación presentamos un ejemplo de la implementación de un esquema Cliente/Servidor para una cajero bancario. El objetivo de este es, el mostrar los elementos que integran un esquema de este tipo, explicar su funcionamiento y las necesidades que se satisfacen con dicha implementación en una entidad bancaria. Se hace referencia al paso correspondiente a la implementación de una aplicación basada en Cliente/Servidor tratada en el capítulo 4.

DETERMINACION DEL ALCANCE

Hemos considerado la estructura de una sucursal bancaria real (BITAL Grupo Financiero), sin embargo, para fines de la aplicación, el universo de servicios se ha reducido, considerando sólo los referentes a Caja y Contabilidad, sin que por ello el objetivo se vea afectado. De tal forma, la estructura operacional de la sucursal está integrada de la siguiente manera :

BACKOFFICE

IDENTIFICACION DE FUNCIONES:

Apertura de:

a) ***Tarjetas de Crédito.***
b) ***Préstamos*** Hipotecarios.
c) ***Préstamos*** para Automóvil.
d) ***Cuentas de Cheques.***
e) Cuentas del ***SAR.***

Cancelaciones de:

a) ***Tarjetas de Crédito.***
b) ***Préstamos*** Hipotecarios.
c) ***Préstamos*** para Automóvil.
d) Cuenta ***SAR.***
e) ***Cuentas de Cheques.***

Aclaraciones de:

a) ***Tarjetas de Crédito.***
b) ***Préstamos*** Hipotecarios.
c) ***Préstamos*** para Automóvil.
d) Cuenta SAR.
e) Cuentas de ***Cheques.***
f) Cualquier otra ***operación bancaria.***

Esta área es cubierta por el personal administrativo de la sucursal (Gerente y Ejecutivos de Cuenta). Es importante señalar que estas operaciones no forman parte de la aplicación, por lo que no se incluyen en el diseño y desarrollo.

IDENTIFICACION DE FUNCIONES:

OPERACIONES

Esta área se encuentra dividida en dos secciones:

- ***CAJA***

Depósitos en:

a) Cuenta de Cheques.
b) ***Inversiones.***

Pagos de:

a) ***Tarjetas de Crédito.***
b) ***Préstamos*** Hipotecarios.
c) ***Préstamos*** para Automóvil.

Ventas de Divisas.

Servicios:

a) Cobro de ***Cheques***
1) ***Cheque*** de otros bancos.
2) ***Cheque*** de nuestro banco.

b) ***Pago*** de luz.
c) ***Pago*** de Teléfono.
d) ***Pago*** de Impuestos.

- ***CONTABILIDAD***

a) Registro y control de las operaciones contables que realiza la sucursal.
b) Emisión de reportes y estadísticas de operación de la sucursal.

IDENTIFICACION DE BENEFICIOS :

Actualización y procesamiento ágil y eficiente.

IDENTIFICACION DE LIMITACIONES:

a) Contabilidad local a la sucursal.
b) No genera estados de cuentas.
c) No genera estados financieros.

Las operaciones correspondientes a Caja y Contabilidad señaladas con anterioridad, con las limitaciones concernientes a esta última; forman la aplicación desarrollada para ejemplificar el funcionamiento de la Arquitectura Cliente/Servidor.

ANALISIS Y DISEÑO

Para el logro de nuestro objetivo consideramos el siguiente esquema Entidad-Relación :

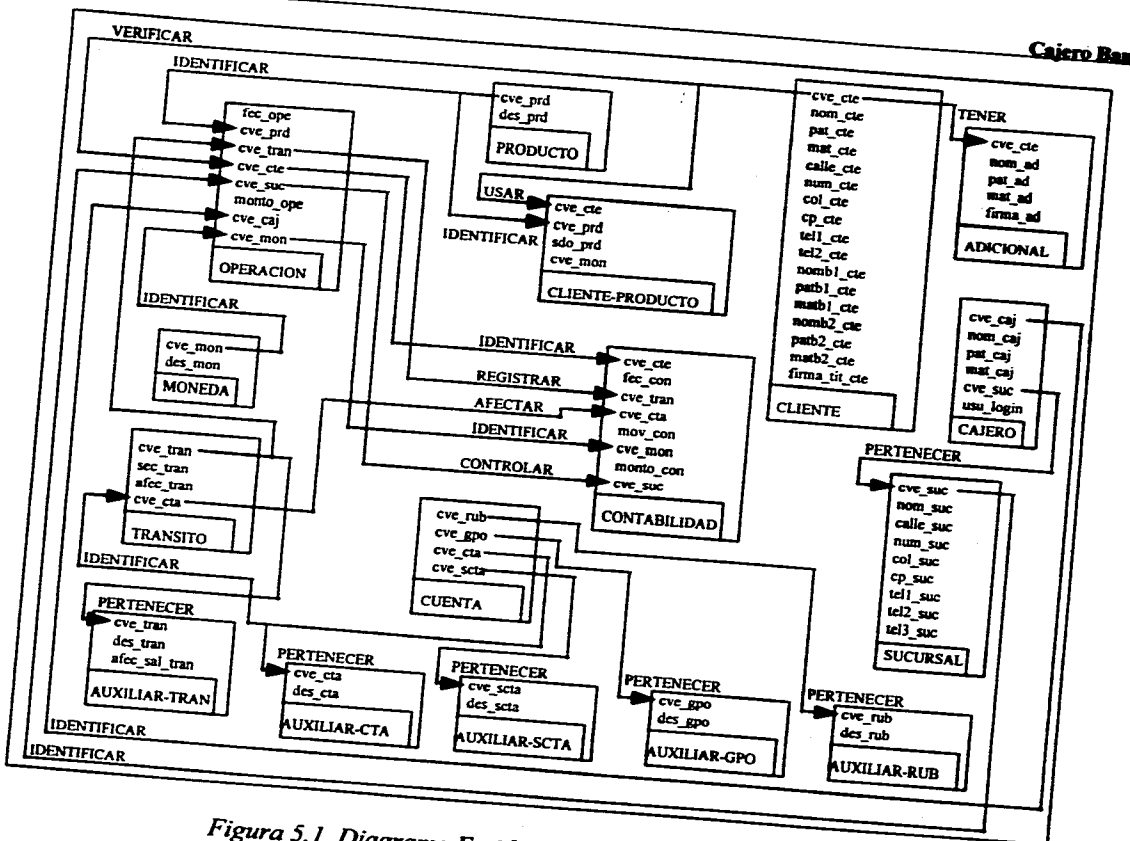


Figura 5.1. Diagrama Entidad Relación Cajero Bancario

DICCIONARIO DE DATOS

El diccionario de Datos generado por tal esquema se muestra a continuación:

ADICIONAL

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_cte	integer	4 *bytes*	Foreignkey	null	clustered	Llave primaria en CLIENTE
nom_ad	char	20		null		Nombre de Cliente Adicional
pat_ad	char	20		null		Apellido paterno Cliente Adicional
mat_ad	char	20		null		Apellido materno Cliente Adicional
firma_ad	image	2Kbytes		null		Firma autorizada Cliente Adicional

AUXILIAR_CTA

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_cta	smallint	2 *bytes*	Primarykey	not null	nonclustered	Número consecutivo de 4 dígitos
des_cta	varchar	60		not null		Nombre de la cuenta

AUXILIAR_GPO

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_gpo	tinyint	1 *byte*	Primarykey	not null	nonclustered	Número consecutivo de 1 dígito (1..6)
des_gpo	varchar	60		not null		Nombre del grupo (Activo, Pasivo, Capital)

AUXILIAR_RUB

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_rub	tinyint	1 *byte*	Primarykey	not null	nonclustered	Número consecutivo de 2 dígitos
des_rub	varchar	60		not null		Nombre del rubro

AUXILIAR_SCTA

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_scta	integer	4 *bytes*	Primarykey	not null	nonclustered	Número consecutivo de 6 dígitos, siendo los primeros 4 los de la cuenta a la que pertenece
des_scta	varchar	60		not null		Nombre de la subcuenta

AUXILIAR_TRAN

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_tran	tinyint	1 *byte*	Primarykey	not null	nonclustered	Número consecutivo
des_tran	varchar	60		not null		Nombre del tránsito

CAJERO

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_caj	tinyint	1 *byte*	Primarykey	not null	clustered	Número consecutivo
nom_caj	char	20		not null		Nombre del cajero
pat_caj	char	20		not null		Apellido Paterno del cajero
mat_caj	char	20		not null		Apellido Materno del cajero
cve_suc	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en SUCURSAL
usu_login	varchar	10		not null		Usuario en el Servidor

CLIENTE

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_cte	integer	4 *bytes*	Primarykey	not null	clustered	Número consecutivo del cliente
nom_cte	char	20		not null		Nombre del cliente
pat_cte	char	20		not null		Apellido Paterno del cliente
mat_cte	char	20		not null		Apellido Materno del cliente
calle_cte	char	20		not null		Calle domicilio del cliente
num_cte	char	10		not null		Número domicilio
col_cte	char	20		not null		Colonia domicilio del cliente
cp_cte	char	5		not null (0.. 9)		Código Postal del domicilio
tel1_cte	char	6		not null		Teléfono domicilio
tel2_cte	char	10		null		Teléfono oficina o celular
nomb1_cte	char	20		not null		Nombre del primer beneficiario
patb1_cte	char	20		not null		Apellido Paterno del primer beneficiario
matb1_cte	char	20		not null		Apellido Materno del primer beneficiario
nomb2_cte	char	20		null		Nombre segundo beneficiario
patb2_cte	char	20		null		Apellido Paterno del segundo beneficiario
matb2_cte	char	20		null		Apellido Materno del segundo beneficiario
firma_tit_cte	image	2 Kbytes		not null		Firma del cliente titular

CLIENTE_PRODUCTO

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_cte	integer	4 *bytes*	Foreignkey	not null	clustered	Llave primaria en CLIENTE
cve_prd	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en PRODUCTO
sdo_prd	money	8 *bytes*		not null		Saldo del Cliente por Producto

CONTABILIDAD

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
fec_con	smallda tetime	4 *bytes*		not null	clustered	Fecha contable del movimiento
cve_suc	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en SUCURSAL
cve_tran	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en AUXILIAR_TRAN
cve_cte	integer	4 *bytes*	Foreignkey	not null		Llave primaria en CLIENTE
cve_cta	smallint	2 *bytes*	Foreignkey	not null		Llave primaria en AUXILIAR_CTA
mov_con	*bit*	1		not null		Tipo de movimiento (0=cargo 1=abono)
impt_con	money	8 *bytes*		not null		Importe de la transacción
cve_mon	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en MONEDA

CUENTA

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_rub	tinyint	1 *byte*	Foreignkey	not null	nonclustered	Llave primaria en AUXILIAR_RUB
cve_gpo	tinyint	1 *byte*	Foreignkey	not null	nonclustred	Llave primaria en AUXILIAR_GPO
cve_cta	smallint	2 *bytes*	Foreignkey	not null	nonclustred	Llave primaria en AUXILIAR_CTA
cve_scta	integer	4 *bytes*	Foreignkey	not null	nonclustered	Llave primaria en AUXILIAR_SCTA

MONEDA

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_mon	tinyint	1 *byte*	Primarykey	not null	nonclustered	Clave del tipo de moneda (nacional o extranjera)
des_mon	varchar	30		not null		Descripción textual del tipo de moneda.

OPERACION

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
fec_ope	smallda tetime	4 *bytes*		not null	clustered	Fecha en la que se registró la transacción
cve_suc	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en SUCURSAL
cve_mon	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en MONEDA
cve_prd	tinyint	1 *byte*	Foreignkey	not null	clustered	Llave primaria en PRODUCTO
cve_cte	integer	4 *byte*	Foreignkey	not null	clustered	Llave primaria en CLIENTE
monto_ope	money	8 *bytes*		not null		Monto de la operación
cve_tran	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en AUXILIAR_TRAN
cve_caj	tinyint	1 *byte*	Foreignkey	not null		Llave primaria en CAJERO

PRODUCTO

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_prd	tinyint	1 *byte*	Primarykey	not null	nonclustered	Número consecutivo
des_prd	varchar	40		not null		Descripción del Producto

SUCURSAL

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_suc	tinyint	1 *byte*	Primarykey	not null	nonclustered	Número consecutivo
nom_suc	char	20		not null		Nombre de la sucursal
calle_suc	char	20		not null		Calle de la dirección de la sucursal
num_suc	char	10		not null		Número de la calle de la sucursal
col_suc	char	20		not null		Colonia de la sucursal
cp_suc	char	5		not null (0.. 9)		Código de la sucursal
tel1_suc	char	6		not null		Teléfono 1 de la sucursal
tel2_suc	char	6		not null		Teléfono 2 de la sucursal
tel3_suc	char	6		not null		Teléfono 3 de la sucursal

TRANSITO

ATRIBUTO	TIPO	TAMAÑO	LLAVE	REGLA	INDICE	DESCRIPCION
cve_tran	tinyint	1 *byte*	Foreignkey	not null	clustered	Llave primaria en AUXILIAR_TRAN
sec_tran	tinyint	1 *byte*		not null		Número secuencial del Tránsito
afec_tran	*bit*	1		not null		Indicador Suma o Resta a la cuenta
cve_cta	smallint	2 *bytes*	Foreignkey	not null		Clave de la cuenta afectada

FASES

DIAGRAMAS DE FLUJO DE DATOS

Los procesos usados se describen con los siguientes Diagramas de Flujo de Datos :

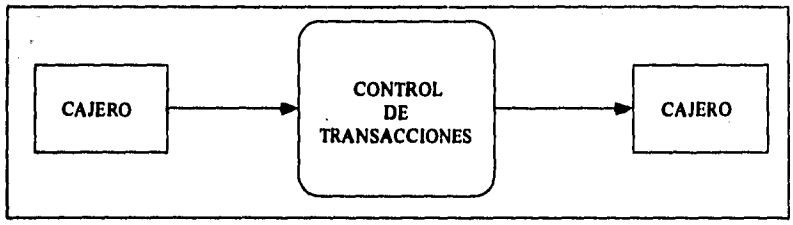


Figura 5.2. Diagrama de Flujo de Datos (DFD) nivel 0

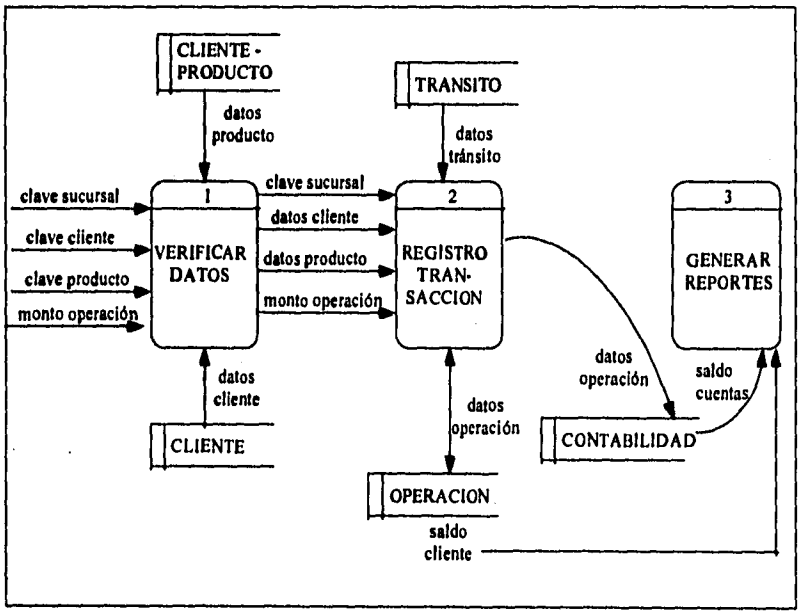


Figura 5.3. Diagrama de Flujo de Datos (DFD) nivel 1

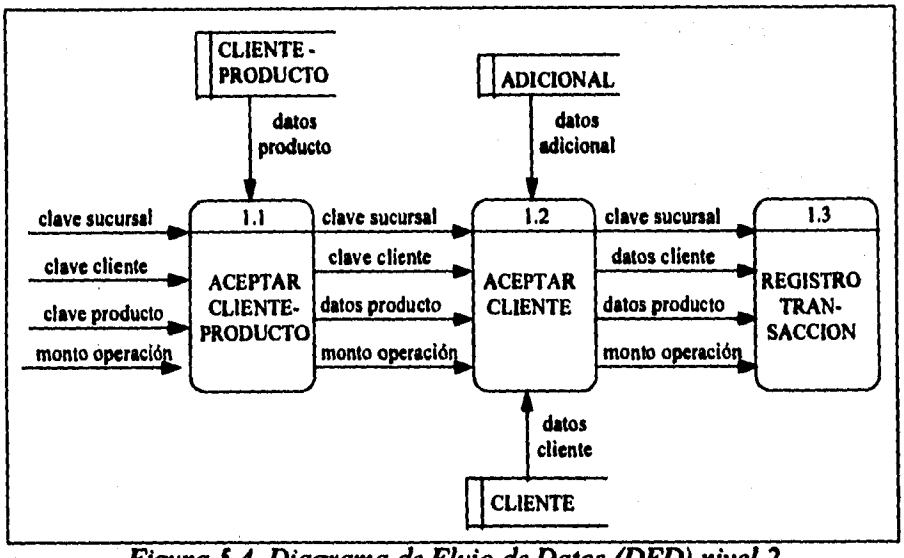


Figura 5.4. Diagrama de Flujo de Datos (DFD) nivel 2

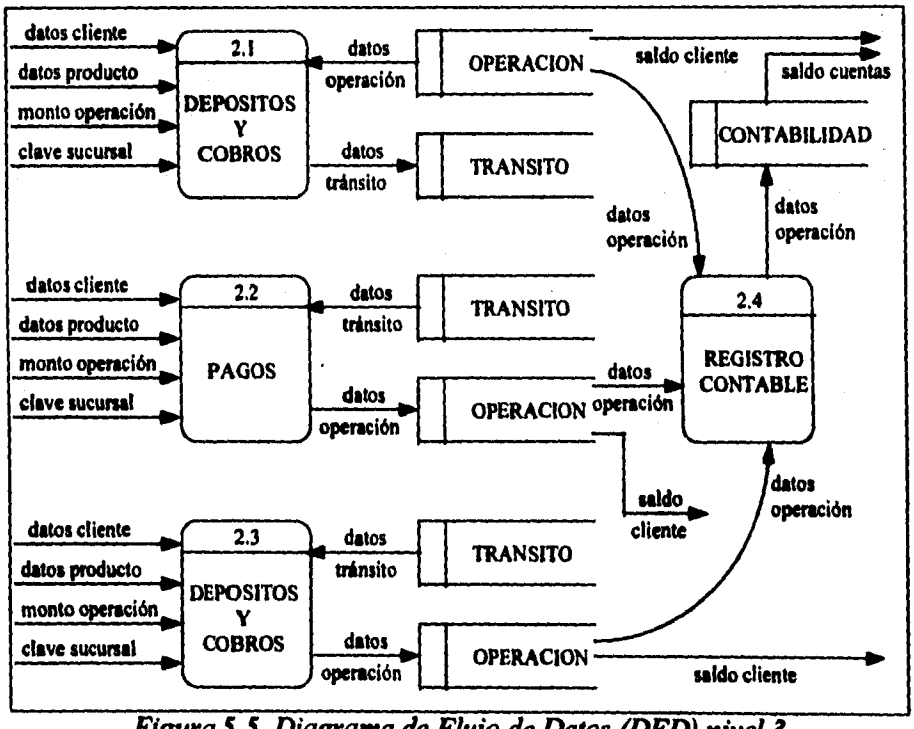


Figura 5.5. Diagrama de Flujo de Datos (DFD) nivel 3

ESQUEMA FISICO Y LOGICO

Por otra parte tanto el diseño físico como el lógico se consideraron de la siguiente manera :

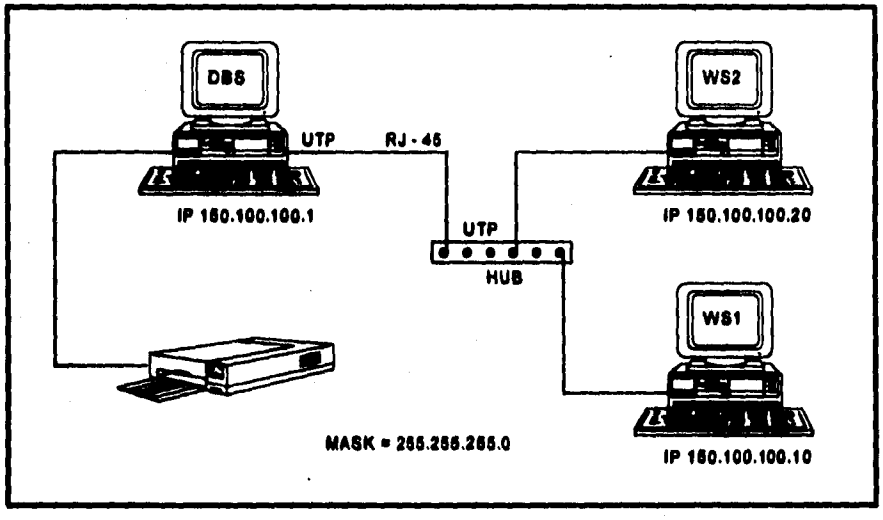


Figura 5.6. Diseño físico.
Red Ethernet Estrella 10Base2 .

La configuración es ***10Base2*** tipo estrella con protocolo de comunicaciones ***TCP/IP***, utilizando cable ***UTP Par trenzado*** y conectores ***RJ45***.

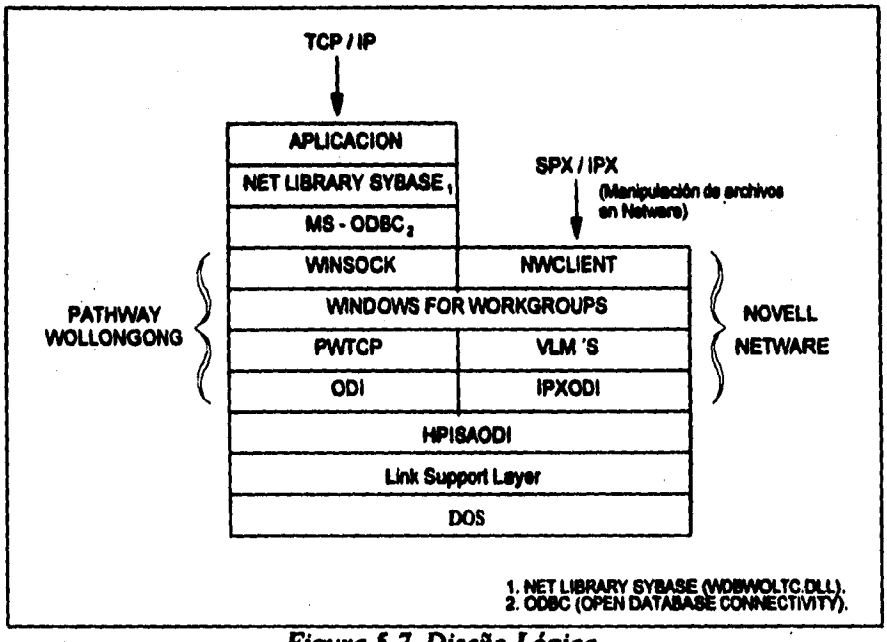


Figura 5.7. Diseño Lógico.
Stack Cliente sobre Open Data Link Interface (ODI)

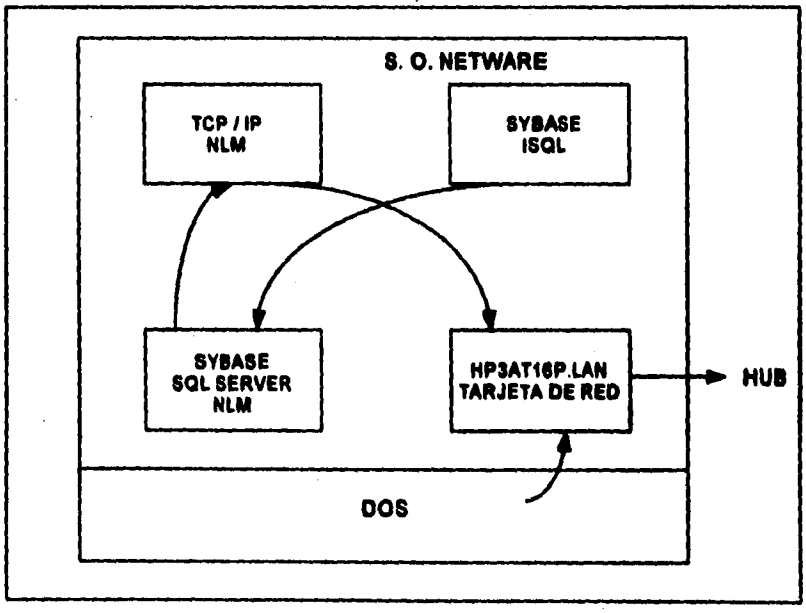


Figura 5.8. Diseño Lógico Servidor.

UBICACION DEL PROCESAMIENTO DE LA INFORMACION

Las características de los elementos empleados para la Arquitectura Cliente/Servidor planteado en la sucursal bancaria son las siguientes:

Servidor

(Servidor local de sucursal)

Hardware :

CPU : Computadora Compatible 486sx, 50 mega ***hertz***.
Tarjeta de red ***Ethernet***.
Disco Duro de 200 mega bytes.
16 mega bytes en ***RAM***.
Monitor Super VGA 14".

Software:

MS DOS Microsoft versión 6.21.
Netware 3.12 de Novell.
Sybase para Novell versión 4.2.2.
TCP/IP NLM (Novell).
Back-End: Procesos de Contabilidad y DBMS.

Cliente (Caja)

Hardware :

CPU : Computadora Compatible 486sx, 50 mega ***hertz***.
Tarjeta de red ***Ethernet***.
Disco Duro de 120 mega bytes.
8 mega bytes en ***RAM***.
Monitor Super VGA 14".

Software:

MS DOS Microsoft versión 6.21
Windows for Work Groups versión 3.11
Visual Basic de Microsoft versión 3.0

Pathway Access Wollongong 3.0
Para acceso a Novell Shell VLM (Novell).
Front-End: Aplicación bancaria y generación de reportes.

Repetidor

Hub ***Ethernet.***
12 puertos ***par trenzado.***
Conector ***RJ45.***
Tarjeta para Cliente ***Ethernet.***

Para el desarrollo del cajero bancario, se utilizaron dos principales herramientas:

1. Sybase para el manejo de bases de datos en el servidor interactuando con servicios de NetWare (***Back-End***).

2. Visual Basic para el desarrollo de las aplicaciones del cliente (***Front End***).

A continuación describiremos estos productos y la forma en que fueron utilizados en el desarrollo de la aplicación.

SYBASE

El Servidor de Sybase ***SQL*** es un producto diseñado para operar en un ambiente de redes de computadoras y que por sus características permite trabajar en un esquema Cliente /Servidor llevando a cabo las siguientes funciones:

- Permite, a través de un lenguaje estructurado de consultas ***SQL***, manejar múltiples bases de datos las cuales incluyen:
  Tablas
  Vistas
  Indices
  Procedimientos

Reglas
Tipos de datos
Defaults
Triggers

- Mantiene un mapeo lógico del almacenamiento físico de los datos.
- Compila y ejecuta transacciones ***SQL***, retornando los resultados al Cliente.
- Reduce considerablemente el tráfico al estar inmerso en un esquema Cliente/Servidor, en comparación con un servidor de archivos de una red LAN. [6]

Con su Lenguaje Estructurado de Consultas (***SQL***) es posible:

- Establecer sentencias de control y flujo de datos.
- Utilizar procedimientos almacenados y ***triggers***.
- Establecer reglas en el manejo de los datos.

En un esquema Cliente/Servidor los clientes establecen conexión con el servidor de Sybase a través de la red, el cual se encarga de administrar los datos requiriendo para esto que el cliente conozca la lógica de la estructura de las bases de datos del Servidor. De esta forma se hace posible que el Cliente reciba sólo el resultado de las transacciones ***SQL*** que le envíe el Servidor optimizando el tiempo de respuesta y disminuyendo el tráfico de la red.

Librerías de Sybase.

Sybase utiliza para su funcionamiento dos librerías principales:

a) La DB-Library- La cual se encarga de manejar la comunicación y la transferencia de datos entre el Servidor SQL y las aplicaciones del Cliente, por medio de una interface para los programas escritos en lenguajes de 3a y 4a generación.

b) La Net-Library- La cual contiene la interface con la red, esta librería permite que cualquier aplicación del cliente, pueda ser

[6] Fast Track to SYBASE, SYBASE Inc.1993 p 1-5

desarrollada independientemente de la red pero que pueda correr a través de ésta.

Un concepto importante dentro de Sybase y de la aplicación creada es el de NLM. Un NLM(NetWare Loadable Module) es una versión especial de un programa que funciona como una extensión del Sistema Operativo NetWare; un NLM puede o no ser cargado dinámicamente, dentro de NetWare, Sybase SQL funciona como un NLM.

Es a través de los programas que se encuentran en estas librerías que podemos interactuar desde el cliente con el servidor, tal es el caso del programa Isql (Interactive sql), el cual nos permite manejar la conexión con el servidor, acepta comandos de transacciones SQL y despliega los resultados con el formato especificado en la aplicación del cliente. Otro programa digno de mencionarse por su utilidad e importancia es el bcp (bulk copy utility), el cual proporciona gran velocidad para el copiado de datos entre las tablas de las bases de datos y un archivo que puede estar en binario o código *ASCII*.

No es objeto de la presente documentación, describir a detalle la instalación y configuración de los productos utilizados en el cajero bancario. Por este motivo nos limitamos a describir sólo los aspectos más importantes de dichos temas, dejando al lector la posibilidad de consultar los manuales referidos en la bibliografía.

Una vez instalado Sybase en el servidor, configuramos los siguientes parámetros dentro del archivo Win.ini para la aplicación:

a) Nombre lógico del Servidor : SERVIDOR.

b) *Dirección IPX* Internet : 51102.

c) Load TCP/IP forward = YES rip = YES

d) name : SEGMENTO_A

Los dos primeros parámetros (a, b) nos permitirán reconocer al servidor por su nombre lógico a través de la red, así como la dirección *IPX* con la que se identificará. El tercer parámetro (c) nos permite *rastrear* a través de los demás servidores (en caso de que existieran). Una vez realizado lo anterior

debemos enlazar y cargar el protocolo que trabaja con el segmento de red (d) que estamos configurando con la siguiente instrucción:

- Bind ipx to SEGMENTO_A net bb51102 addr 150.100.100.1

Desde el indicador :

: load isql - Usa -P

con esto cargamos el Isql entrando como super usuario (Usa).

Para crear el directorio donde residirá la aplicación es necesario crear un device, que es como Sybase reconoce el espacio físico que se asignará a la base de datos con los siguientes comandos :

```
1> disk init name = "devsuc"
2> physame = "sys: sybase/sybdev/devsuc.dat"
3> vdevno = 8, size = 20480
4>go
```

En estos comandos asignamos el nombre (devsuc) y tamaño de nuestro ***device***(20,480 páginas de 512 kb, lo cual ocupa un total aproximado de 40 mbs), así como el número de dispositivo que representa para Sybase (el 8 en este caso).

Acto seguido, podemos crear ya nuestra base de datos, asignando 35 mb para la misma y 5 mb para la bitácora (LOG) con el siguiente comando:

```
1>create database SUCURSAL on DEVSUC = 35
  log on DEVSUC = 5
2>go
```

Para la creación de grupos y cuentas de usuarios:

```
1> sp_addgroup sucursal
2> go
1> sp_add login blanca
2> go
```

VISUAL BASIC

Ya que usamos Visual Basic como herramienta de la aplicación cajero bancario, consideramos importante mencionar una breve historia sobre el cómo surge y el porqué es utilizado para este tipo de sistemas.

En 1964 la programación en computadoras era tediosa pues se requería invertir gran cantidad de tiempo. En un esfuerzo para facilitar esta programación John G. Kemeny y Thomas E. Kurtz, profesores de Dartmouth, desarrollaron el lenguaje BASIC.

BASIC (Beginner's All-Purpose Symbolic Instruction Code), a diferencia de otros lenguajes que se desarrollaron al mismo tiempo, era interactivo. Es decir, que en el momento en que se introducía un comando, BASIC debía verificarlo e indicar si es que se había cometido un error, lo cual permitió aprender el lenguaje rápidamente e iniciar una revolución en la programación de computadoras, ya que en los otros lenguajes primero se tenía que teclear todo el programa y si se cometía algún error éstos no lo identificaban hasta que se terminaba, cuando esto ocurría el programador tenía que encontrarlos.

Ya que BASIC era amigable y fácil de usar varias compañías de computadoras lo ofrecían, por ejemplo las computadoras PC 's de IBM tenían una versión llamada BASICA. Las computadoras compatibles con IBM pronto ofrecieron una versión similar llamada GW-BASIC. Incluso la versión 5.0 de MS-DOS incluyó una nueva versión llamada QBASIC. Sin embargo, ya que BASIC fué diseñado originalmente para principiantes, los programadores avanzados lo consideraban como un lenguaje de "juguete".

Las cosas cambiaron cuando Microsoft introdujo Microsoft Windows. Ahora los programadores no sólo tenían que escribir programas para trabajar, también tenían que escribir programas que ofrecieran menús en cascada, ventanas de diálogo y con facilidad de navegar en éstas, esencialmente la programación ahora tiene dos trabajos.

En 1991 Microsoft realizó Visual Basic. Así los programadores pueden primero diseñar la apariencia de sus programas y después concentrarse en los programas de trabajo. Es decir que ya no se tiene que invertir tanto

tiempo en el diseño de pantallas amigables pues Visual Basic facilita esta parte.

Características

Visual Basic 3.0 de Microsoft es un lenguaje orientado a eventos con herramientas para la construcción de GUI's. Las librerías de Visual Basic están construidas sobre el estandar SQL Server de Microsoft las cuales facilitan la creación de diferentes tipos de aplicaciones basadas en Windows incluyendo Front-Ends.

A través de Visual Basic se pueden crear objetos gráficos (botones de comandos, botones de opciones, Paneles, frames, gráficas de diversos estilos, Cuadros de Diálogos, etc.) que facilitan al usuario el manejo de diversas aplicaciones; estos objetos conservan propiedades que permiten ajustarse a las necesidades del usuario. Visual Basic también permite intercambio dinámico de datos (DDE) e interface para el manejo de documentos múltiples (MDI).

CONCLUSIONES

1. Cliente/Servidor implica romper con esquemas de trabajo tradicionales en la empresa, por lo que ésta debe estar sensibilizada para el cambio con el fin de obtener ventajas competitivas.

2. Implementar una Arquitectura Cliente/Servidor implica un alto costo, por lo cual debe llevarse a cabo un estudio detallado de los beneficios que se alcanzan con esta arquitectura en comparación con los recursos económicos, humanos y materiales con que debe contar una organización que desee trabajar con Cliente/Servidor.

3. Debe existir un compromiso entre el área de Sistemas y los usuarios para poder implementar exitosamente un Arquitectura Cliente/Servidor.

4. El punto más crítico de la implementación de una Arquitectura Cliente/Servidor es la integración de la interoperabilidad de las aplicaciones existentes y futuras de la empresa.

5. Se deben llevar a cabo labores de análisis y diseño continuo de procesos para la implementación de una arquitectura Cliente/Servidor.

6. Para implementar una Arquitectura Cliente/Servidor se requiere de un programa adecuado de capacitación del personal de Sistemas debido a la gran interoperabilidad que existe entre componentes de hardware y software.

7. La aplicación del Cajero Bancario es un modelo de implementación de Cliente/Servidor en dos partes, en la cual se ejemplifica la distribución del procesamiento de la información.

8. La Arquitectura Cliente/Servidor en tres partes la recomendamos, en base a lo antes mencionado, para empresas medianas y grandes cuyas operaciones y transacciones requieran procesamiento distribuido.

9. Debido a la proliferación y dominio público del protocolo de comunicaciones TCP/IP se ha convertido en uno de los elementos más importantes para lograr la interoperabilidad entre las diversas plataformas y arquitecturas de cómputo.

10. Es necesario invertir gran cantidad de recursos humanos y económicos en el diseño y construcción de las interfaces adecuadas para la integración de los sistemas Cliente/Servidor.

11. Cliente/Servidor permite trabajar con elementos heterogéneos de hardware y software.

APENDICE A

CONFIGURACION DE LA RED

```
REM ===== Configuración del Servidor Sybase ==========================
REM
REM Autores: Camacho Moreno Marco Antonio
REM          De la Rosa Castilla René Saúl
REM          López Jiménez Blanca Graciela
REM Tesis :  Cliente/Servidor un Auténtico Cambio en Tecnología Informática
REM          y su Aplicación en un Cajero Bancario.
REM
REM ================================================================

file server name SERVIDOR
ipx internal net 51102

REM Carga los Servicios de TCP/IP
load tcpip forward=yes rip=yes
load snmp
load snmplog

REM ========== Carga Tarjeta de Red ============================
load hp3at16p int=3 port=300 name=SEGMENTO_A frame=ethernet_II
bind ipx to SEGMENTO_A net=bb51102
bind ip to SEGMENTO_A addr=150.100.100.1 mask=255.255.255.0
mount all

REM ========== Carga de Sybase Sql Server V.4.2.2 ===========
search add sys:sybase/nlms
load directfs
load c:\server.312\sybstubs
load sqlsrvr
#
REM === Archivo de Interfaces para el Sybase SQL Server, usando TCP/IP y
SPX/IPX====
#
SYBASE
#  tcp port number is 1000 (hex) or 4096 (decimal)
#  ip address is 150.100.100.1 (decimal)
        query tli tcp /dev/tcp \x020010009664640100000000000000000
        master tli tcp /dev/tcp \x020010009664640100000000000000000
        console tli tcp /dev/tcp \x020010019664640100000000000000000
#  spx port number is 83bd (hex) or 33725 (decimal)
#  network number is 51102 (hex) or 332034 (decimal)
        query tli spx /dev/nspx \x0005110200000000000183bd
        master tli spx /dev/nspx \x0005110200000000000183bd
        console tli spx /dev/nspx \x0005110200000000000183be
#
```

```
#
REM=== Variables de ambiente del Servidor Sybase SQL Server para Novell Netware
====

SYBASE = SYS:/SYBASE/
SYBASE_DMASTER = SYS:/SYBASE/MASTER.DAT
SYBASE_NLMS = SYS:/SYBASE/NLMS/
SYBASE_INTERFAC = SYS:/SYBASE/INTERFAC
SYBASE_LOG = SYS:/SYBASE/ERRORLOG
DSQUERY = SYBASE
DSLISTEN = SYBASE
DSCONSOLE = SYBASE
EDITOR = edit
USE_DEFAULT_SPX = FALSE

#*****************************************************************
REM
REM          Configuración de la Estación de Trabajo (Cliente)
REM
#*****************************************************************
#
REM ================ Autoexec.bat ================
@ECHO OFF
LH C:\DOS\MSCDEX.EXE /S /V /D:MSCD001 /L:F /M:15
SET SOUND=C:\SBPRO
SET BLASTER=A220 I7 D1 T4
SET VIDEOBLST=C:\VBLASTER A2AF0 I5
lh C:\SBPRO\SBP-SET /M:12 /VOC:12 /CD:12 /FM:12
PROMPT $P$G
set PATH=C:\DOS;C:\WINDOWS;C:\windows\red;C:\;C:\MOUSE;d:\uti\QEDIT;
SET TEMP=C:\WINDOWS\TEMP
LH C:\DOS\DOSKEY.COM
LH C:\DOS\keyb sp,,c:\dos\keyboard.sys
LH MOUSE.COM
LH C:\DOS\SHARE.EXE /L:800 /F:500
rem ================== Configuración de Red ================
lh c:\windows\red\lsl.com
lh c:\windows\red\hpisaodi.com
lh C:\WINDOWS\net start
lh C:\WINDOWS\odihlp.exe
rem ================================================================
rem #---------------------- The Wollongong Group ---------------------#
rem # Las siguientes líneas son de configuración para la ruta del Runtime

SET PATH=D:\AP\PATHWAY;%PATH%
```

```
PWCONFIG -N:65
ODI -I:f
PWTCP


REM================== CONFIG.SYS=======================
DEVICE=C:\DOS\SETVER.EXE
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /e:2048 /p
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\EMM386.EXE NOEMS HIGHSCAN
DOS=HIGH,UMB
DEVICE=C:\DOS\SMARTDRV.EXE 1024 1024
FILES=90
BUFFERS=40,0
STACKS=9,256
rem DEVICEHIGH=C:\MOUSE\MOUSE.SYS
DEVICEHIGH=C:\SBPRO\DRV\SBPCD.SYS /D:MSCD001 /P:220
LASTDRIVE=Z
DEVICE=C:\WINDOWS\IFSHLP.SYS


REM ============ Tabla del Host para PathWay ======================
#
# Formato:  Dirección IP formada por cuatro octetos seguidos  por un nombre de dominio
# específico y opcionalmente seguido por un alias.
# Ejemplo:
#
#129.84.3.243
#

REM ============ Archivo de Hosts en la Red de la Tesis ============
150.100.100.1 Servidor Banco_Bital
150.100.100.10 Estacion_1 Blanca
150.100.100.20 Estacion_2 Rene
150.100.100.30 Estacion_3 Marco


REM====== Archivo NET.CFG de Configuración de Red ==============
# Acceso a la Red Novell  y a la Red TCP/IP usando Pathway Access 3.0

#---------- The Wollongong Group ----------#
# Las siguientes líneas indican la configuración de la
# interface del  PathWay  para ODI
```

```
Protocol IP
        Bind HPISAODI
Protocol ARP
        Bind HPISAODI
#---------- The Wollongong Group ----------#

link driver HPISAODI
  Frame Ethernet_802.3
  Frame Ethernet_802.2
  Frame Ethernet_SNAP
  PROTOCOL IPX 8137 ETHERNET_II
   Frame Ethernet_II
   INT 15
#---------- The Wollongong Group ----------#
# Las siguientes líneas indican la configuración de la
# interface del PathWay para ODI

        frame ETHERNET_II
        Protocol IP 000000000800 ETHERNET_II
        Protocol ARP 000000000806 ETHERNET_II
#---------- The Wollongong Group ----------#

Link Support
  Max Boards 4
  Buffers  8 1500
  MemPool  4096

#---------- Uso de VLM's ----------#
Netware Dos Requester
  READ ONLY COMPATIBILITY=OFF
  FIRST NETWORK DRIVE = G
  USE DEFAULTS = OFF
  VLM = CONN.VLM
  VLM = IPXNCP.VLM
  VLM = TRAN.VLM
  VLM = SECURITY.VLM
  VLM = BIND.VLM
  VLM = NWP.VLM
  VLM = FIO.VLM
  VLM = PRINT.VLM
  VLM = GENERAL.VLM
  VLM = REDIR.VLM
  VLM = NETX.VLM
  VLM = NDS.VLM
  VLM = RSA.VLM
```

APENDICE

B

ESQUEMA DE LA BASE DE DATOS EN SYBASE SQL SERVER

(CODIGO FUENTE)

========== Esquema de la Base de Datos en Sybase SQL Server ==========

Autores: Camacho Moreno Marco Antonio
De la Rosa Castilla René Saúl
López Jiménez Blanca Graciela
Tesis : Cliente/Servidor un Auténtico Cambio en Tecnología Informática y su Aplicación en un Cajero Bancario.

==========

```
EXEC sp_adduser blanca,'blanca','public'
GO
EXEC sp_adduser rene,'rene','public'
GO
EXEC sp_adduser marco,'marco','public'
GO

CREATE TABLE dbo.ADICIONAL (cve_cte int NULL,
                nom_ad char(20) NULL,
                pat_ad char(20) NULL,
                mat_ad char(20) NULL,
                firma_ad image NULL)
GO
CREATE TABLE dbo.AUXILIAR_CTA (cve_cta smallint NOT NULL,
                des_cta varchar(60) NOT NULL)
GO
CREATE TABLE dbo.AUXILIAR_GPO (cve_gpo tinyint NOT NULL,
                des_gpo varchar(60) NOT NULL)
GO
CREATE TABLE dbo.AUXILIAR_RUB (cve_rub tinyint NOT NULL,
                des_rub varchar(60) NOT NULL)
GO
CREATE TABLE dbo.AUXILIAR_SCTA (cve_scta int NOT NULL,
                des_scta varchar(60) NOT NULL)
GO
CREATE TABLE dbo.AUXILIAR_TRAN (afec_sal_tran char(1) NOT NULL,
                cve_tran tinyint NOT NULL,
                des_tran varchar(60) NOT NULL)
GO
```

```
CREATE TABLE dbo.CAJERO (cve_caj tinyint NOT NULL,
            nom_caj char(20) NOT NULL,
            pat_caj char(20) NOT NULL,
            mat_caj char(20) NOT NULL,
            cve_suc tinyint NOT NULL,
            usu_login varchar(10) NOT NULL)
GO
CREATE TABLE dbo.CLIENTE (cve_cte int NOT NULL,
            nom_cte char(20) NOT NULL,
            pat_cte char(20) NOT NULL,
            mat_cte char(20) NOT NULL,
            calle_cte char(30) NOT NULL,
            num_cte char(10) NOT NULL,
            col_cte char(30) NOT NULL,
            cp_cte char(5) NOT NULL,
            tel1_cte char(6) NOT NULL,
            tel2_cte char(10) NULL,
            nomb1_cte char(20) NOT NULL,
            patb1_cte char(20) NOT NULL,
            matb1_cte char(20) NOT NULL,
            nomb2_cte char(20) NULL,
            patb2_cte char(20) NULL,
            matb2_cte char(20) NULL,
            firma_tit_cte image NULL)
GO
CREATE TABLE dbo.CLIENTE_PRODUCTO (cve_mon tinyint NOT NULL,
                  cve_cte int NOT NULL,
                  cve_prd tinyint NOT NULL,
                  sdo_prd money NOT NULL)
GO
CREATE TABLE dbo.CONTABILIDAD (fec_con smalldatetime NOT NULL,
                  cve_suc tinyint NOT NULL,
                  cve_cte int NOT NULL,
                  cve_cta tinyint NOT NULL,
                  mov_con bit NOT NULL,
                  monto_con money NOT NULL,
                  cve_mon tinyint NOT NULL,
                  cve_tran tinyint NOT NULL)
GO
CREATE TABLE dbo.CUENTA (cve_cta smallint NOT NULL,
            cve_gpo tinyint NOT NULL,
            cve_rub tinyint NOT NULL,
            cve_scta int NOT NULL)
GO
```

```
CREATE TABLE dbo.MONEDA (cve_mon tinyint NOT NULL,
            des_mon varchar(30) NOT NULL)
GO
CREATE TABLE dbo.OPERACION (fec_ope smalldatetime NOT NULL,
                cve_suc tinyint NOT NULL,
                cve_mon tinyint NOT NULL,
                cve_prd tinyint NOT NULL,
                cve_cte int NOT NULL,
                cve_caj tinyint NOT NULL,
                cve_tran tinyint NOT NULL,
                monto_ope money NOT NULL)
GO
CREATE TABLE dbo.PRODUCTO (cve_prd tinyint NOT NULL,
            des_prd varchar(40) NOT NULL)
GO
CREATE TABLE dbo.SUCURSAL (cve_suc tinyint NOT NULL,
            nom_suc char(20) NOT NULL,
            calle_suc char(20) NOT NULL,
            num_suc char(10) NOT NULL,
            col_suc char(20) NOT NULL,
            cp_suc char(5) NOT NULL,
            tel1_suc char(6) NOT NULL,
            tel2_suc char(6) NOT NULL,
            tel3_suc char(6) NOT NULL)
GO
CREATE TABLE dbo.TRANSITO (cve_tran tinyint NOT NULL,
            sec_tran tinyint NOT NULL,
            afec_tran bit NOT NULL,
            cve_cta smallint NOT NULL)
GO

CREATE unique nonclustered INDEX sucursal ON SUCURSAL( cve_suc)
 GO
CREATE unique nonclustered INDEX moneda ON MONEDA( cve_mon)
 GO
CREATE unique nonclustered INDEX producto ON PRODUCTO( cve_prd)
 GO
CREATE clustered INDEX contabilidad ON CONTABILIDAD( fec_con)
 GO
CREATE clustered INDEX operacion ON OPERACION(
 fec_ope,cve_cte,cve_prd)
 GO
CREATE clustered INDEX transito ON TRANSITO(
 cve_tran)
 GO
```

```
CREATE nonclustered INDEX aux_cta ON AUXILIAR_CTA(
 cve_cta)
 GO
CREATE nonclustered INDEX aux_gpo ON AUXILIAR_GPO(
 cve_gpo)
 GO
CREATE unique nonclustered INDEX aux_rub ON AUXILIAR_RUB(
 cve_rub)
 GO
CREATE unique nonclustered INDEX aux_scta ON AUXILIAR_SCTA(
 cve_scta)
 GO
CREATE clustered INDEX cliente ON CLIENTE(
 cve_cte)
 GO
CREATE nonclustered INDEX cuenta ON CUENTA(
 cve_cta,cve_gpo,cve_rub,cve_scta)
 GO
CREATE unique nonclustered INDEX aux_tran ON AUXILIAR_TRAN(
 cve_tran)
 GO
CREATE clustered INDEX cte_prd ON CLIENTE_PRODUCTO(
 cve_cte,cve_prd)
 GO
CREATE clustered INDEX cajero ON CAJERO(
 cve_caj)
 GO
CREATE clustered INDEX adicional ON ADICIONAL(
 cve_cte)
 GO

SETUSER 'dbo'
GO
GRANT Delete ON ADICIONAL TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON AUXILIAR_CTA TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON AUXILIAR_GPO TO public
GO
SETUSER 'dbo'
GO
```

```
GRANT Delete ON AUXILIAR_RUB TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON AUXILIAR_SCTA TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON AUXILIAR_TRAN TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON CAJERO TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON CLIENTE TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON CLIENTE_PRODUCTO TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON CONTABILIDAD TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON CUENTA TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON MONEDA TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON OPERACION TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON PRODUCTO TO public
GO
SETUSER 'dbo'
GO
```

```
GRANT Delete ON SUCURSAL TO public
GO
SETUSER 'dbo'
GO
GRANT Delete ON TRANSITO TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON ADICIONAL TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON AUXILIAR_CTA TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON AUXILIAR_GPO TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON AUXILIAR_RUB TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON AUXILIAR_SCTA TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON AUXILIAR_TRAN TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON CAJERO TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON CLIENTE TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON CLIENTE_PRODUCTO TO public
GO
SETUSER 'dbo'
GO
```

```
GRANT Insert ON CONTABILIDAD TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON CUENTA TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON MONEDA TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON OPERACION TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON PRODUCTO TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON SUCURSAL TO public
GO
SETUSER 'dbo'
GO
GRANT Insert ON TRANSITO TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON ADICIONAL TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON AUXILIAR_CTA TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON AUXILIAR_GPO TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON AUXILIAR_RUB TO public
GO
SETUSER 'dbo'
GO
```

```
GRANT Select ON AUXILIAR_SCTA TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON AUXILIAR_TRAN TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON CAJERO TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON CLIENTE TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON CLIENTE_PRODUCTO TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON CONTABILIDAD TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON CUENTA TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON MONEDA TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON OPERACION TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON PRODUCTO TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON SUCURSAL TO public
GO
SETUSER 'dbo'
GO
```

```
GRANT Select ON TRANSITO TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON sysalternates TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON syscolumns TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON syscomments TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON sysdepends TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON sysindexes TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON syskeys TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON syslogs TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON sysobjects TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON sysprocedures TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON sysprotects TO public
GO
SETUSER 'dbo'
GO
```

```
GRANT Select ON syssegments TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON systypes TO public
GO
SETUSER 'dbo'
GO
GRANT Select ON sysusers TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON ADICIONAL TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON AUXILIAR_CTA TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON AUXILIAR_GPO TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON AUXILIAR_RUB TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON AUXILIAR_SCTA TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON AUXILIAR_TRAN TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON CAJERO TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON CLIENTE TO public
GO
SETUSER 'dbo'
GO
```

```
GRANT Update ON CLIENTE_PRODUCTO TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON CONTABILIDAD TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON CUENTA TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON MONEDA TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON OPERACION TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON PRODUCTO TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON SUCURSAL TO public
GO
SETUSER 'dbo'
GO
GRANT Update ON TRANSITO TO public
GO

sp_primarykey SUCURSAL,cve_suc
GO
sp_primarykey MONEDA,cve_mon
GO
sp_primarykey PRODUCTO,cve_prd
GO
sp_primarykey AUXILIAR_CTA,cve_cta
GO
sp_primarykey AUXILIAR_GPO,cve_gpo
GO
sp_primarykey AUXILIAR_RUB,cve_rub
GO
sp_primarykey AUXILIAR_SCTA,cve_scta
GO
```

```
sp_primarykey CLIENTE,cve_cte
GO
sp_foreignkey CONTABILIDAD,SUCURSAL,cve_suc
GO
sp_foreignkey CONTABILIDAD,MONEDA,cve_mon
GO
sp_foreignkey OPERACION,SUCURSAL,cve_suc
GO
sp_foreignkey OPERACION,MONEDA,cve_mon
GO
sp_foreignkey OPERACION,PRODUCTO,cve_prd
GO
sp_foreignkey CUENTA,AUXILIAR_CTA,cve_cta
GO
sp_foreignkey CUENTA,AUXILIAR_GPO,cve_gpo
GO
sp_foreignkey CUENTA,AUXILIAR_RUB,cve_rub
GO
sp_foreignkey CUENTA,AUXILIAR_SCTA,cve_scta
GO
```

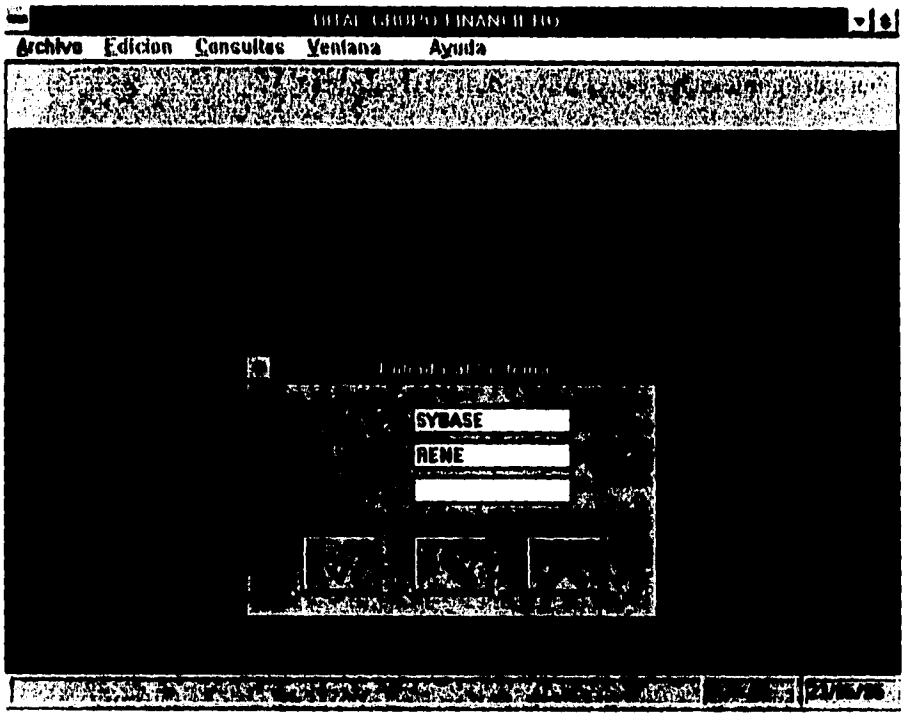
APENDICE C

CODIGO DE LA APLICACION

```
rem Autores: Camacho Moreno Marco Antonio
rem          De la Rosa Castilla René Saúl
rem          López Jiménez Blanca Graciela
remTesis :   Cliente/Servidor un Auténtico Cambio en Tecnología Informática
rem          y su Aplicación en un Cajero Bancario.


rem Variables para manipular la conexión con el Sybase SQL Server

Global DBLIB_VERSION$
Global PrimaryWindowTitle$
Global SqlConn%
Global ServerName$
Global LoginID$
Global Password$
Global DatabaseName$
Global SQLStatus%
Global Const LoginTimeout% = 30
Global Const QueryTimeout% = 60
Global Const ProgramName$ = "BANCO BITAL"

rem Declaraciones Globales para el DB-Library de Visual Basic .
rem usadas por los api's con el DLL VBSQL (vbsql.vbx)

Global Const SUCCEED% = 1
Global Const FAIL% = 0

Global Const INTEXIT% = 0
Global Const INTCONTINUE% = 1
Global Const INTCANCEL% = 2

Global Const MOREROWS = -1
Global Const NOMOREROWS = -2
Global Const REGROW = -1
Global Const BUFFULL = -3

Global DBLIB_VERSION$
Global PrimaryWindowTitle$
Global SqlConn%
Global ServerName$
Global LoginID$
Global Password$
Global DatabaseName$
Global SQLStatus%
```

```
Global Const QueryTimeout% = 60
Global Const ProgramName$ = "BANCO BITAL"

rem Declaraciones Globales para el DB-Library de Visual Basic .
rem usadas por los api's con el DLL VBSQL (vbsql.vbx)

Global Const SUCCEED% = 1
Global Const FAIL% = 0

Global Const INTEXIT% = 0
Global Const INTCONTINUE% = 1
Global Const INTCANCEL% = 2

Global Const MOREROWS = -1
Global Const NOMOREROWS = -2
Global Const REGROW = -1
Global Const BUFFULL = -3

rem Códigos de estado para dbresults(). Los valores posibles son
rem SUCCEED, FAIL, y NO_MORE_RESULTS.

Global Const NOMORERESULTS = 2
```

Pantalla de acceso al sistema

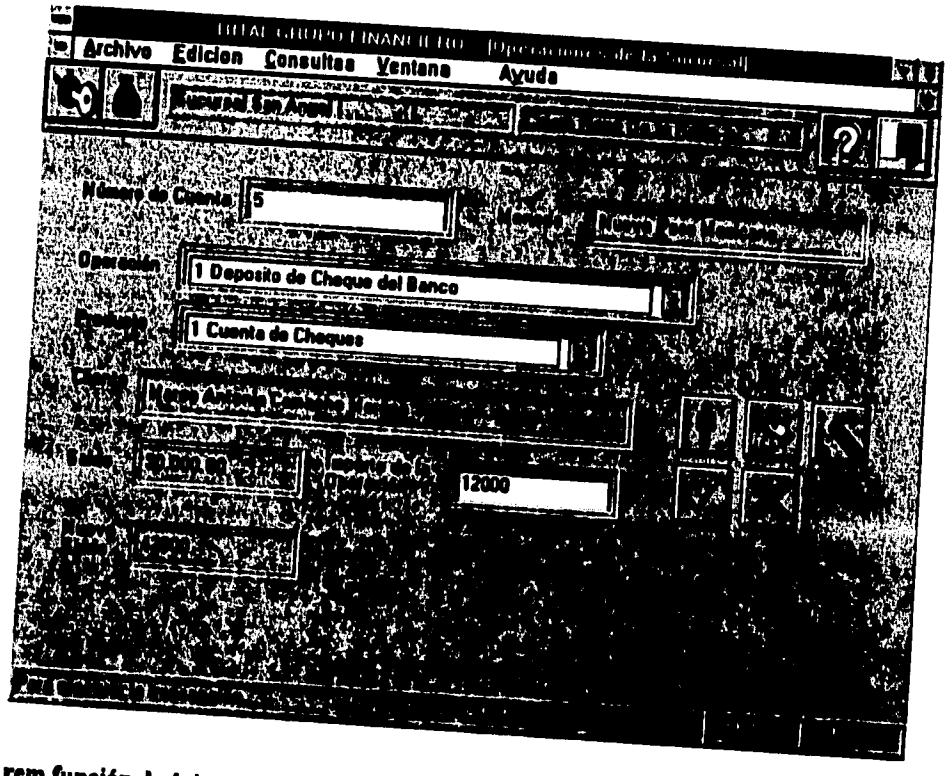



```
Sub baceptar_Click ()
  Dim ok%
inicial.MousePointer = RELOJ
If Len(tservidor.Text) > 0 And Len(tusuario.Text) > 0 Then
  servername$ = Trim(UCase(tservidor.Text))
  loginid$ = Trim(LCase(tusuario.Text))
  password$ = Trim(LCase(tclave.Text))
  ok% = loginToserver()    ' Hace conexión al servidor
  inicial.MousePointer = FLECHA
  If Not ok% = SUCCEED Then
    MsgBox "Usuario erróneo o Falla al conectarse con el servidor, reintente "
  Else
    Unload Me
    'Checa el número de Cajero y el número de la Sucursal
    query$ = "select SUCURSAL.nom_suc,CAJERO.nom_caj,CAJERO.pat_caj from
    SUCURSAL,CAJERO "
    query$ = query$ + "where CAJERO.usu_login = " + "'" + Trim(loginid$) + "'"
    query$ = query$ + " and SUCURSAL.cve_suc = CAJERO.cve_suc "
    MsgBox "Este es el query:" + query$ + ";"
```

```
  resultado% = sqlcmd%(SqlConn%, query$)
  resultado% = sqlexec%(SqlConn%)
  total_resultados% = SqlResults%(SqlConn%)
  If sqlrows%(SqlConn%) = FAIL Or total_resultados% = FAIL Then
    MsgBox "No existe usuario, o problemas de conexión"
  Else
    Do Until SqlResults%(SqlConn%) = NOMORERESULTS
      Do Until SqlNextRow%(SqlConn%) = NOMOREROWS
        inicial.sucursal.Caption = "Sucursal " + Trim(CStr(sqldata(SqlConn%, 1)))
        inicial.cajero.Caption = "Cajero " + Trim(CStr(sqldata(SqlConn%, 2))) + " " +
        Trim(CStr(sqldata(SqlConn%, 3)))
      Loop
    Loop
  End If
  Call habilita_todo
  operaciones.Show
 End If
Else
 MsgBox "Favor de llenar los campos"
End If
End Sub

rem función que establece conexión con el servidor

Function loginToserver () As Integer

loginToserver = SUCCEED
If SqlConn% <> 0 Then SqlClose (SqlConn%)
Status% = SqlSetLoginTime%(LoginTimeout%)
SqlConn% = SqlOpenConnection(Servername$, LoginID$, Password$, ProgramName$,
ProgramName$)
If SqlConn% <> 0 Then
  DatabaseName$ = SqlName(SqlConn%)
  ChangePrimaryWindowCaption
  Result% = SqlSetTime%(QueryTimeout%)
Else
  DatabaseName$ = ""
  Servername$ = ""
  loginToserver = FAIL
End If
End Function
```

```
rem verifica la conexión existente

Function CheckServerConnection (SqlConn As Integer) As Integer
  If SqlConn% <> 0 Then
    CheckServerConnection = 1
  Else
    CheckServerConnection = 0
  End If
End Function

rem función que extrae la información de los tránsitos

Sub Form_Load ()
' obtención de la información de los tránsitos
  query$ = "select cve_tran,des_tran from AUXILIAR_TRAN order by cve_tran"
  resultado% = sqlcmd%(SqlConn%, query$)
  resultado% = sqlexec%(SqlConn%)
  total_resultados% = SqlResults%(SqlConn%)
  If sqlrows%(SqlConn%) = FAIL Or total_resultados% = FAIL Then
    MsgBox "No hay información sobre tránsitos, o problemas de conexión"
  Else
    Do Until SqlResults%(SqlConn%) = NOMORERESULTS
      Do Until SqlNextRow%(SqlConn%) = NOMOREROWS
        lista_tran.AddItem Trim(CStr(sqldata(SqlConn%, 1))) + " " +
Trim(CStr(sqldata(SqlConn%, 2)))
      Loop
    Loop
  End If
  ' Obtención de la información de los productos
  query$ = "select cve_prd,des_prd from PRODUCTO order by cve_prd"
  resultado% = sqlcmd%(SqlConn%, query$)
  resultado% = sqlexec%(SqlConn%)
  total_resultados% = SqlResults%(SqlConn%)
  If sqlrows%(SqlConn%) = FAIL Or total_resultados% = FAIL Then
    MsgBox "No hay Información de los Productos, o problemas de conexión"
  Else
    Do Until SqlResults%(SqlConn%) = NOMORERESULTS
      Do Until SqlNextRow%(SqlConn%) = NOMOREROWS
        lista_prod.AddItem Trim(CStr(sqldata(SqlConn%, 1))) + " " +
Trim(CStr(sqldata(SqlConn%, 2)))
      Loop
    Loop
  End If
End Sub
```

Pantalla de captura de datos para una operación de depósito




```
rem función de éxito o fallo del comando

Function ExecuteSQLCommand (cmd As String) As Integer

SQLStatus% = SUCCEED
ExecuteSQLCommand = SUCCEED
If SqlCmd(SqlConn%, cmd$) = FAIL% Then
   SQLStatus% = FAIL
   ExecuteSQLCommand = FAIL
End If
If SqlExec(SqlConn%) = FAIL% Then
   SQLStatus% = FAIL
   ExecuteSQLCommand = FAIL
End If
End Function
```

```
rem  proceso que trae la información del cliente

Sub baceptar_Click ()

  If Len(porbuscar.Text) = 0 Then
     MsgBox "Recuerda llenar el número de Cuenta..."
   Else
     query$ = "select nom_cte,pat_cte,mat_cte from CLIENTE where cve_cte = " +
        CStr(Trim(porbuscar.Text))
     'MsgBox "Este es el query:" + query$ + ":"
     resultado% = sqlcmd%(SqlConn%, query$)
     resultado% = sqlexec%(SqlConn%)
     total_resultados% = SqlResults%(SqlConn%)
     If sqlrows%(SqlConn%) = FAIL Or total_resultados% = FAIL Then
      MsgBox "No existe el cliente, o problemas de conexión"
     Else
       Do Until SqlResults%(SqlConn%) = NOMORERESULTS
        Do Until SqlNextRow%(SqlConn%) = NOMOREROWS
         Cliente.Caption = Trim(CStr(sqldata(SqlConn%, 1))) + " " +
        Trim(CStr(sqldata(SqlConn%, 2))) + " " + Trim(CStr(sqldata(SqlConn%, 3)))
        Loop
       Loop
     End If
     'Trae el saldo de CLIENTE-PRODUCTO
     query$ = "select CLIENTE_PRODUCTO.sdo_prd, MONEDA.des_mon from
CLIENTE_PRODUCTO, MONEDA where CLIENTE_PRODUCTO.cve_cte = " +
Trim(CStr(porbuscar.Text)) + " and CLIENTE_PRODUCTO.cve_prd = " +
CStr(Mid(lista_prod.Text, 1, 1)) + " and CLIENTE_PRODUCTO.cve_mon =
MONEDA.cve_mon "
     'MsgBox "Este es el query:" + query$ + ":"
     resultado% = sqlcmd%(SqlConn%, query$)
     resultado% = sqlexec%(SqlConn%)
     total_resultados% = SqlResults%(SqlConn%)
     If sqlrows%(SqlConn%) = FAIL Or total_resultados% = FAIL Then
      MsgBox "El cliente no tiene dicho producto, o problemas de conexión"
     Else
       Do Until SqlResults%(SqlConn%) = NOMORERESULTS
        Do Until SqlNextRow%(SqlConn%) = NOMOREROWS
         saldo.Caption = Trim(CStr(sqldata(SqlConn%, 1)))
         moneda.Caption = Trim(CStr(sqldata(SqlConn%, 2)))
        Loop
       Loop
     End If
   End If
```

```
End Sub

rem proceso que afecta la base de datos
Sub Command3D1_Click ()

ReDim arr_afec(10) As String
ReDim arr_cta(10) As String
ReDim arr_sec(10) As String

   'Búsqueda del Signo para afectar la cuenta
   query$ = "select afec_sal_tran from AUXILIAR_TRAN where cve_tran = " +
Mid(Trim(lista_tran.Text), 1, 1)
   resultado% = sqlcmd%(sqlconn%, query$)
   resultado% = sqlexec%(sqlconn%)
   total_resultados% = SqlResults%(sqlconn%)
   If sqlrows%(sqlconn%) = FAIL Or total_resultados% = FAIL Then
     MsgBox "No existe el tránsito, o problemas de conexión"
   Else
     Do Until SqlResults%(sqlconn%) = NOMORERESULTS
      Do Until SqlNextRow%(sqlconn%) = NOMOREROWS
        signo$ = Trim(CStr(sqldata(sqlconn%, 1)))
      Loop
     Loop
     saldo_num& = CLng(saldo.Caption)
     impte_num& = CLng(importe.Text)
     If signo$ = "+" Then
       rdo& = saldo_num& + impte_num&
     Else
       rdo& = saldo_num& - impte_num&
     End If
     nusaldo.Caption = CStr(rdo&)

     'Actualización del Saldo
     query$ = "BEGIN TRAN "        'Inicia transacción
     resultado% = sqlcmd%(sqlconn%, query$)
     resultado% = sqlexec%(sqlconn%)
     Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
       Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
       Loop
     Loop

     'Ejecuta update
     query$ = "update CLIENTE_PRODUCTO set CLIENTE_PRODUCTO.sdo_prd = "
     + CStr(nusaldo.Caption)
     query$ = query$ + " where cve_cte = " + Trim(CStr(porbuscar.Text))
```

```
query$ = query$ + " and cve_prd = " + Trim(Mid(CStr(lista_prod.Text), 1, 1))
resultado% = sqlcmd%(sqlconn%, query$)
resultado% = sqlexec%(sqlconn%)
Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
  Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
  Loop
Loop
status& = Sqlcount&(sqlconn%)  'Obtiene el número de renglones afectados
todook% = 1
If status& = FAIL Then        'Hubo problemas haz rollback de la transacción
  MsgBox "No se hizo la actualización en CLIENTE_PRODUCTO, problemas "
  query$ = "ROLLBACK TRAN"
  todook% = 0
  resultado% = sqlcmd%(sqlconn%, query$)
  resultado% = sqlexec%(sqlconn%)
  Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
    Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
    Loop
  Loop
End If
'Afectación en la tabla de la Operación
If todook% = 1 Then                'Se asegura de que actualizó
    hoy$ = Format(Date, "mm/dd/yy")
    cve_prod$ = Mid(lista_prod.Text, 1, 1)
    cve_tran$ = Mid(lista_tran.Text, 1, 1)
    cve_cte$ = Trim(porbuscar.Text)

    'Inicia la ejecución del Insert
    query$ = "insert OPERACION values ("
    query$ = query$ + "'" + hoy$ + "'" + "," + gbl_sucursal + "," + gbl_moneda
    query$ = query$ + "," + cve_prod$ + "," + cve_cte$ + "," + gbl_cajero + ","
    query$ = query$ + cve_tran$ + "," + Trim(importe.Text) + ")"

    MsgBox "Query es:" + query$ + ":"
    resultado% = sqlcmd%(sqlconn%, query$)
    resultado% = sqlexec%(sqlconn%)
    Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
      Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
      Loop
    Loop
    todobien% = 1
    status& = Sqlcount&(sqlconn%)  'Obtiene el número de renglones afectados
    If status& = FAIL Then        'Hubo problemas haz rollback de la transaceión
      MsgBox "Problemas en actualización de Operación, no se efectuó!"
      todobien% = 0
```

```
    query$ = "ROLLBACK TRAN"
    resultado% = sqlcmd%(sqlconn%, query$)
    resultado% = sqlexec%(sqlconn%)
    Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
     Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
     Loop
    Loop
  End If
  'Hace actualización a la Contabilidad
  If todobien% = 1 Then   'Se asegura de que actualizó
    'Trae los datos de tránsito para afectar la contabilidad
    query$ = "select cve_cta,afec_tran,sec_tran from TRANSITO"
    query$ = query$ + " where cve_tran = " + cve_tran$ + " order by sec_tran"
    resultado% = sqlcmd%(sqlconn%, query$)
    resultado% = sqlexec%(sqlconn%)
    ind% = 0
    Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
     Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
       arr_cta(ind%) = Trim(CStr(sqldata(sqlconn%, 1)))
       arr_afec(ind%) = Trim(CStr(sqldata(sqlconn%, 2)))
       arr_sec(ind%) = Trim(CStr(sqldata(sqlconn%, 3)))
       ind% = ind% + 1
     Loop
    Loop
    salte% = 0
    'Inserta cíclicamente las cuentas en contabilidad
    For x% = 0 To ind% - 1
      query$ = "insert CONTABILIDAD values (" + "'" + hoy$ + "'" + ","
      query$ = query$ + gbl_sucursal + "," + cve_cte$ + "," + "'" + arr_cta(x%) +
      "'" + ","
      query$ = query$ + arr_afec(x%) + "," + Trim(importe.Text) + ","
      query$ = query$ + gbl_moneda + "," + cve_tran$ + ")"
      MsgBox "Query es:" + query$ + ":"
      resultado% = sqlcmd%(sqlconn%, query$)
      resultado% = sqlexec%(sqlconn%)
      Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
       Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
       Loop
      Loop
      status& = Sqlcount&(sqlconn%)  'Obtiene el número de renglones afectados
      If status& = FAIL Then         'Hubo problemas haz rollback de la transacción
       MsgBox "Problemas en actualización de Contabilidad, no se efectuo!"
       query$ = "ROLLBACK TRAN"
       salte% = 1                    'Forza a no continuar los inserts
      End If
```

```
            If salte% = 1 Then
              Exit For      'Sale del ciclo para no hacer actualización
            End If
          Next x%
          If salte% = 0 Then              ' Hace el Commit siempre y
            query$ = "COMMIT TRAN"        ' cuando no haya un rollback
          End If
          resultado% = sqlcmd%(sqlconn%, query$)
          resultado% = sqlexec%(sqlconn%)
          Do While (SqlResults%(sqlconn%) <> NOMORERESULTS)
            Do While (SqlNextRow(sqlconn%) <> NOMOREROWS)
            Loop
          Loop
        End If
      End If
    End If
End Sub



rem función de salida de la aplicación

Sub exitapplication ()

rem Exit SQL, then exit the application.
  SqlExit
  SqlWinExit
  End
End Sub

rem  subrutina que cierra la conexión

Sub logoff ()
  If SqlConn% <> 0 Then
    SqlClose (SqlConn%)
    Servername$ = "[No server]"
    DatabaseName$ = "[no database]"
    ChangePrimaryWindowCaption
  End If
End Sub
```

REPORTE EMITIDO (Arqueo de Caja)

BITAL GRUPO FINANCIERO [Arqueo de Caja]	
Sucursal: SAN ANGEL I Cajero: RENE DE LA ROSA	
Moneda: Nuevo Peso Mexicano Fecha: 28/04/95	
SALDO INICIAL 10,000.00 SALDO FINAL 22,850.00	
Depósito cuenta de cheques	25,500.00
Pago de Préstamo de Automóvil	1,500.00
Pago de Luz	150.00
Depósito cuenta de cheques	3,000.00
Depósito en Inversión a 30 días	15,700.00
Pago de Préstamo Hipotecario a más de 20 años	1,000.00
Depósito cuenta de cheques	6,000.00
Cobro de cheque	(10,000.00)
Cobro de Inversión a 30 días	(30,000.00)

GLOSARIOS

- TERMINOS INFORMATICOS
- DEL NEGOCIO

TERMINOS INFORMATICOS

a

Ancho de Banda

La diferencia entre la frecuencia más alta y la más baja de un canal de transmisión, expresada en Hertz (Hertz = Ciclos por segundo). Una medida de la capacidad de información de un canal de transmisión. El ancho de banda varía de acuerdo al tipo y método de transmisión.

ANSI

Abreviación de "American National Standars Institute". Una institución que ayuda a definir estándares, y que también representa a los E.U. en la Organización Internacional de Estándares (ISO).

API

Siglas de "Application Program Interface". En general, todo el grupo de funciones o procedimientos, que se invocan desde un programa de aplicación, para utilizar un software de base. Por ejemplo: APIs para OS/2, APIs para cierto gateway, etc.

APPC

Siglas de "Advanced Program to Program Communication". APPC es un protocolo "puerto-a-puerto", definido por IBM (y ahora también parte de SAA). No está restringido a micros, ni a equipos IBM. Define un conjunto de verbos (mapeados y básicos) para que dos dispositivos puedan lograr una "conversación" en la cual no existe una jerarquía maestro-esclavo. Existen ya diversas implementaciones de APPC para micros. Bajo el léxico IBM, para que un dispositivo sea capaz de "hablar" APPC, debe tener una categoría de unidad lógica 6.2 (LU6.2) por lo que frecuentemente ambos términos son usados como sinónimos.

APPN

Siglas de "advanced Peer-to-Peer Networking". Arquitectura de red de acuerdo con IBM que provee conexión punto-a-punto entre

computadoras. Bajo APPN, no se requiere un host. Implementa conceptos tales como directorios de red dinámicos y enrutamientos dinámicos en SNA.

Archivo Plano

Conjunto de registros que no cuentan con una estructura definida.

ASCII

Siglas de "American Standard Code for Information Interchange". Forma estándar de codificar los caracteres en un patrón de 7 bits. El ASCII extendido utiliza 8 bits y logra codificar 256 patrones (2^8), en lugar de 128 (2^7).

b

Backbone

Generalmente se denomina de esta manera a la conexión entre redes locales.

Back-End

En general software o hardware que actúa sin ser visto en una arquitectura cliente-servidor. En un manejador de Bases de Datos (DBMS) se denomina así a la parte del software, ubicada en el Servidor, que se encarga de seleccionar, controlar, ordenar, indexar y administrar la información. Ver Front-End.

Batch

Un método de procesamiento de datos en donde todos los trabajos se agrupan primero para después enviarse, en forma secuencial a la computadora para su proceso.

Bit

Un dígito binario; puede ser 0 o 1. Es la unidad más pequeña de información que indica dos estados "off(0), on(1)".

Boot

Proceso. de carga de los programas básicos para encender la computadora. Bajo el léxico IBM, IPL (Initial Program Load).

Boot Remoto

En una red, proceso de encender una estación de trabajo, haciendo el "boot" desde el servidor de la red.

Bps

Abreviación de bits por segundo. La medida de velocidad de transmisión más utilizada. En redes locales lo más frecuente es hablar de Mbps (Mega bits por segundo). Es importante hacer notar que la abreviación de bit es una b minúscula, mientras que la de Byte es una B mayúscula.

Byte

Conjunto de 8 bits que representan un caracter en binario.

C

Cache, Caching

En computadoras muy rápidas, la memoria cache tiene como objetivo suministrarle los datos al procesador a la velocidad que los solicita (sin retrasos). Para tal efecto, dado que la memoria cache es de menor tamaño que el RAM ordinario, trata de "saber" que datos son los más usados y tenerlos disponibles para el procesador, (El porcentaje de aciertos se le llama Hit-Ratio). Por similitud, hacer "caching" de disco, es la tarea de tener en RAM los sectores más utilizados de disco, agilizando de esta manera su acceso.

Canal

Un camino físico o lógico que permite la transmisión de información. En algunos casos puede ser sinónimo de Bus.

CASE

Siglas de "Computer Aided Software Engineering". La utilización de software para ayudar en la definición, elaboración, designación, documentación y algunas otras áreas de desarrollo de programas.

CCITT

Siglas de "Comité Consultivo Internacional de Telegrafía y Telefonía". Fija estándares internacionales en comunicaciones. Se encuentra ubicado en Ginebra, Suiza.

CDS

Servicios de Directorio de Celdas

CPU

Siglas de "Central Processing Unit". Generalmente se utiliza este término para definir el Procesador Central de una computadora. Es la base de una computadora digital.

CSMA/CD

Siglas de "Carrier Sense Multiple Access/Collision Detection". Técnica utilizada para enviar señales dentro de una red local. El cable se utiliza por "competencia", y cuando una tarjeta detecta solo la portadora, empieza a trasmitir, pero debe seguir escuchando por si ocurre alguna colisión. De ser así, requiere hacer una retransmisión.

d

Data link, Nivel de

Nivel 2 del modelo OSI. En este nivel se arman los "frames" y se verifican errores de transmisión (usualmente a través de código CRC).

DBMS

Siglas de "Data Base Management System". Sistema Manejador de Base de Datos

DB2

Manejador de bases de datos de IBM para ambientes MVS (mainframes). Utiliza SQL, y define en sí mismo un dialecto estándar.

Desktop

Microcomputadora.

Device

Dispositivo de entrada y/o salida.

DFS

Siglas de "Distributed File System". Sistema distribuido de archivos, para poder accesar desde un equipo X, los achivos de otro equipo Y. Creado por SUN Microsystems.

Dirección

Un conjunto de números que identifican de manera única "algo". Puede ser una estación de trabajo en una red, una localidad de memoria, un paquete de datos viajando en una red, una tarjeta de red, etc.

DLL

Siglas de "Data Link Library".

DNS

Servicio de Nombre de Dominio.

Downsizing

El proceso conocido como downsizing consiste en emigrar aplicaciones completas o funciones de éstas, de mainframe centralizados a redes o sistemas centralizados menores, buscando así acercar la aplicación al usuario final. Toma ese nombre de las técnicas de "adelgazamiento" de los sistemas modernos de producción.

Driver

Manejador. Es un conjunto de rutinas de software que se utilizan para controlar el intercambio de información entre un dispositivo y el CPU.

DTS

Servicios Distribuidos de Temporalización.

e

EBCDIC

Siglas de "Extended Binary Coded Decimal Interchange Code". Método de IBM para codificar caracteres en una forma binaria.

Encriptamiento

Proceso donde los datos de un mensaje, por seguridad, son codificados para protegerlos de accesos no deseados.

Estándar

Especificación que debe utilizar un sistema para cumplir con las características que exige el mercado si es que quiere ser compatible y lograr la comunicación.

Equipo carente de disco

Estación de trabajo que no posee diskettes, ni discos duros y que por lo tanto, hace un "boot remoto" (Diskless Workstation).

Estación de trabajo

Cualquier equipo conectado a una red, con capacidad propia de proceso.

Estación remota

En general, nombre que se le da a las PCs que se conectan a una red local a través de modem.

Ethernet

El estándar de tarjetas de red más conocido y sólido. Define una velocidad de transmisión de 10 Mbits/seg, utilizando protocolo *CSMA/CD*.

f

FAX

Texto o gráficas transmitidas vía líneas de comunicación a un punto remoto donde un original es reproducido. La transmisión puede ser analógica o digital. Existen tarjetas para integrar este servicio a una red local.

FDDI

Siglas de "Feber Distributed Data Interface". El estándar para transmisión de datos en redes locales utilizando fibra óptica, a una velocidad de 100 mbps. Utiliza un doble anillo en una topología similar a Token-Ring, incluso en la definición del frame. Igualmente utiliza protocolo de Token-Passing para control de la RED.

Físico, nivel

Primer nivel del modelo OSI. Define las características del medio de transmisión (cable en la mayoría de los casos), velocidad, forma de codificar los bits, etc.

Front-End

En ambiente de bases de datos, el software que le presenta la información al usuario (Reside en la estación de trabajo).

FTP

Siglas de "File Transfer Protocol". Un servicio de alto nivel bajo ambiente TCP (Ver TCP/IP) que permite y controla el proceso de transferencia de archivos a través de una red.

g

Gateway

Dispositivo que permite conectar dos redes (locales o geográficamente distantes) con diferentes protocolos. Un gateway cambia al menos, los protocolos de los primeros 4 niveles del modelo ISO/OSI.

GDS

Servicio de Directorio Global.

GUI

Siglas de "Graphical User Interface". Enlace de comunicación o interfaz entre un usuario y el sistema operativo de una computadora. Generalmente utiliza pantallas diseñadas con base en iconos (figuras) que representan las funciones disponibles para el Usuario, Windows de Microsoft, es un ejemplo de un GUI.

h

Hardware

Equipo físico. Todos los componentes electrónicos y mecánicos de una red, como computadoras personales, tarjetas de red y cables.

Hertz (Hz)

Unidad de Frecuencia, equivalente a un ciclo por segundo.

Host

Comúnmente sinónimo de mainframe. En la terminología es cualquier nodo en una interred, capaz de brindar algún servicio.

Hub

Normalmente asociado a 10 Base T. En ese sentido, un hub inteligente se define como aquel que tiene un agente ***SNMP***.

i

IAB

Siglas de "Internet Activity Board". Comité independiente de investigadores y profesionales con intereses técnicos para el buen desempeño del sistema Internet.

IEEE

Siglas de "Institute of Electrical and Electronic Engineers". Instituto de profesionistas que se encarga de crear, promover y soportar especificaciones y estándares de comunicaciones. El comité 802 del IEEE ha definido diversos estándares para redes locales.

IEEE-802.3

Basado en Ethernet, define una forma de protocolo basada en CSMA/CD. El estándar 802.3 tiene diversas variantes (cable grueso, delgado, par trenzado y broadband).

IEEE-802.5

Define un tipo de hardware "Token-Ring". Aunque IBM patrocinó gran parte de este comité, en última instancia, el Token-Ring que IBM lanza al mercado, es un superconjunto del 802.5.

Integridad de Entidad

Regla por la que cada entidad en un archivo, debe ser reconocida de manera única. Un buen manejador de base de datos (DBMS) debe observar esta regla.

Integridad Referencial

Regla por la cual se garantiza que; en el caso de que cualquier dato dentro de una entidad, haga referencia a (sea llave de) otra entidad en otra tabla, esta última entidad siempre existirá. En resumen, no se permite hacer referencia a un registro que no existe en el otro archivo.

Integridad

Característica de la información de reflejar datos congruentes con la realidad.

Interoperabilidad

Proceso donde las computadoras pueden operar interactuando con otras a través de una red sin conversión de datos o intervención humana.

IP

Siglas de "Internet Protocol". En la familia TCP/IP, IP es el encargado de definir la mejor ruta y enviar por ésta los paquetes, en una

comunicación sin conexión (connectionless). Es decir, IP en si mismo, no garantiza la recepción correcta de paquetes, ni su ordenamiento correcto.

IPX

Protocolo "puerto-a-puerto", propio de Novell, que actúa en el nivel 3 del Modelo OSI (Nivel de Red). Entre sus ventajas está el tener direcciones de tres campos: nodo, red y socket, que le permiten tener enlaces entre redes y varios procesos corriendo en diferentes servidores. Está basado en el protocolo de nivel 3 de XNS.

ISDN

Siglas de "Integrated Services Digital Network". Red digital de servicios integrados. Estándar que define una línea digital telefónica, con canales para voz y datos.

ISO

Siglas de "International Standards Organization". Institución internacional que se encarga de especificar estándares en diversas áreas.

j

Job

Conjunto de instrucciones

k

Kerberos

Sistema de seguridad desarrollado en MIT el cual otorga autenticidad a los usuarios. No da acceso a servicios o base de datos sino que establece identidad al logon, el cual es utilizado durante una determinada sesión.

Kernel

Parte del Sistema Operativo que interactúa directamente con el hardware.

l

LAN

Siglas de "Local Area Network". La abreviación más común al hablar de Redes de Area Local.

Login

Acción de entrar a utilizar un host o un servidor de red, establecer una sesión de trabajo y ser reconocido como usuario por el Sistema Operativo.

m

Mainframe

Unidad principal: Suele utilizarse para identificar una computadora "grande" (capaz de ser la estructura principal de una instalación de informática).

Mouse

Dispositivo de entrada que funciona con interfaces gráficas.

Multitarea

La capacidad de un sistema operativo de realizar más de una tarea en forma simultánea. OS/2, por ejemplo, es un sistema operativo que brinda capacidades de multitasking.

Multiusuario

La capacidad de un sistema operativo de atender a más de un usuario en forma simultánea. UNIX, por ejemplo, es un sistema operativo que brinda capacidades de multiusuario.

MVS

Siglas de "Multiple Virtual Storage". Sistema operativo de IBM, el cual optimiza operaciones en línea, tiempo real, multiusuario y multitarea.

n

Named Pipes

Mecanismo nativo de Lan-Manager, para brindar comunicación entre procesos (IPC) y diversos nodos, facilitando el procesamiento distribuido.

NetBios

Interfaz estándar para comunicar dos estaciones de trabajo en una red local. Definido por IBM y Sytek en 1984-1985. Dentro del contexto de MS-DOS, son los servicios de software y firmware que implementan la interfaz entre las aplicaciones y la tarjeta de red.

NetView

Producto de software desarrollado por IBM que permite controlar redes complejas cómo aquellas que se forman utilizando SNA y Redes Locales. NetView solamente puede operar con productos de red definidos por IBM.

Nodo

Este término se utiliza generalmente para referirse a una estación de trabajo dentro de una red. Punto computacional dentro de una red de comunicaciones.

o

OLE

Objetos Ligados que permiten pasar de una aplicación a otra sin abandonar la primera.

OS/2

Sistema Operativo desarrollado por IBM-Microsoft para la línea de sistemas personales PS/2 (Personal System/2).

OSF

Siglas de "Open Software Foundation". Organización de proveedores de soluciones para UNIX, encargada de estandarizar este mercado.

OSI

Siglas de "Open Systems Interconnect". Estructura lógica y estándar de 7 niveles de protocolos definida por ISO para facilitar la comunicación en ambientes heterogéneos.

p

Peer-to-peer

Una comunicación peer-to-peer (Puerto-a-Puerto) se establece cuando las dos computadoras pueden iniciar una conversación y no requiere de "permiso" de la otra.

Presentación, Nivel de

El nivel 6 del Modelo OSI. Sus funciones principales son realizar labores de "transformación" de la información. Conversión de formatos (v.g. de ASCII a EBCDIC), encripción y/o compresión.

Procedimientos Almacenados (Stored Procedure)

Conjunto de comandos SQL almacenados en la base de datos que pueden ser ejecutados por nombre. Estos procedimientos pueden aceptar y regresar parámetros, valores y llamar a otros procedimientos. Los stored procedures corren más rápido que los mismos comandos ejecutados interactivamente o en batch.

r

RAM

Siglas de "Random Access Memory". Memoria que puede ser escrita y leída de manera dinámica. Puede ser accesada por el usuario en cualquier punto con facilidad y sin tener que leer grabaciones anteriores.

Red, Nivel de

El tercer nivel en el Modelo OSI. Su función es cambiar las referencias de nombres de nodos, a direcciones de los mismos, y definir la ruta a tomar.

RED Local

Conjunto de computadoras, enlazadas por algún tipo de cable, y en distancias relativamente cercanas (dentro de un mismo edificio o campus). También conocida como LAN (Local Area Network).

RISC

Siglas de "Reduced Instruction Set Computing".

RJ45

Conector para cable de par trenzado (UTP y STP).

ROM

Siglas de "Read-Only Memory". Memoria no-volátil que puede ser leída pero no modificada.

RUTEADOR

Dispositivo que toma un paquete (Nivel 3 del Modelo OSI) y lo envía del punto A al punto B, después de analizar cual es el camino óptimo para llegar a su destino. Esto se logra gracias a la información que cada ruteador almacena sobre todos los nodos de la red.

RPC

Siglas de "Remote Procedure Call". El proceso utilizado en ambiente UNIX con TCP/IP para implementar un proceso específico en un nodo local o remoto.

S

SDK

Siglas de "Software Development Kit". Paquete de desarrollo de aplicaciones.

SDLC

Siglas de "Synchronous Data Link Control". Protocolo de nivel 2 del Modelo OSI, estándar en la arquitectura SNA de IBM. Se utiliza principalmente para transmisiones punto-a-punto.

Sesión, Nivel de

Nivel 5 del modelo OSI. Su función es establecer la conexión entre los dos extremos de la conversación.

Servidor de Impresión

Equipo (puede ser una PC) enfocada a atender las colas de espera para las impresoras conectadas a éste. Un Servidor de Impresión es útil cuando deseamos compartir impresoras diferentes de aquellas que están conectadas al Servidor de la RED.

Sistemas Biométricos

Son aquellos sistemas de seguridad que identifican al usuario por medio de signos biológicos como la voz, huellas digitales, etc.

SNA

Siglas de "Systems Network Arquitecture". La arquitectura de protocolos para redes creada por IBM.

SNMP

Siglas de "Simple Network Management Protocol". Protocolo estándar de la familia TCP/IP, enfocado al manejo, administración y control de redes que utilicen TCP/IP.

SOCKET

Un socket es un objeto abstracto desde el cual los datos son transferidos. Los sockets existen dentro de un dominio de comunicación y cambian datos con otros sockets residiendo en el mismo dominio.

Software

Son aquellos programas de computadora, estructuras de datos y documentación asociada que sirven para realizar el método lógico, procedimiento o control requerido.

SQL-Base

Servidor de Bases de Datos creado por Gupta Technologies. El primero que surgió en el mercado (1986).

SQL-Server

Servidor de Bases de Datos desarrollado por Microsoft y Sybase (hasta 1989 fué comercializado por Ashton-Tate). Se liberó al mercado apenas en mayo de 1989. Posee características sumamente poderosas en manejo de transacciones, integridad de la información y control de concurrencia.

SQL-Windows

Front-end muy poderoso desarrollado por Gupta Technologies. Es una herramienta tipo 4GL, para generar programas bajo Windows, que accesan algún motor SQL (v.g. SQL-Base, SQL-Server, Oracle).

SQL

Siglas de "Structured Query Language". El lenguaje de consulta y acceso a base de datos más común en la actualidad. Definido como estándar por IBM, ANSI e ISO.

t

TCP/IP

Juego de protocolos creados en los 70s por Vince Cerf, profesor de Stanford, por encargo del Pentágono. El objetivo era lograr protocolos independientes del hardware. Hoy en día, son los protocolos que permiten la mayor conectividad entre los más diversos equipos (desde una Mac, hasta una Cray).

TCP

Transmission Control Protocol. Nivel 4 de la familia TCP/IP. TCP es un protocolo orientado a conexiones, que garantiza la llegada de paquetes y su ordenamiento.

TELNET

Servicio de terminal virtual especificado por el Departamento de Defensa de E. U. e implementado en la mayoría de versiones de UNIX.

TFTP

Siglas de "Trivial File Transfer Protocol". Protocolo de transferencia de archivos basados en UNIX. TFTP es una simplificación de SFTP.

Transacción

El término transacción se define como un conjunto de operaciones enmarcadas por un comienzo y un final.

Transporte, Nivel de

El cuarto nivel del modelo OSI. Sus principales funciones son secuenciar paquetes, y verificar si han llegados todos.

Trigger

Es un tipo especial de procedimiento almacenado, el cual se encuentra ligado a una tabla. Los triggers son activados automáticamente por el servidor SQL cuando en una tabla se insertan, modifican o borran datos (insert, update o delete). Un trigger puede detectar los eventos que se están realizando en ese momento, esto si en la tabla se insertó, modificó o borro un dato.

Twisted-Pair (Par Trenzado)

Cable que se forma de dos alambres aislados, que se tuercen entre sí (de ahí el nombre de par trenzado). Existen dos variantes básicas: Blindado y no-blindado. El blindado permite mayores distancias y es mucho más inmune al ruido. El no-blindado (o UTP) es más económico, pero tiene limitantes de distancias y ruido. En los E. U. ha tenido mucha difusión en los últimos meses, debido en primera instancia a que el costo de mano de obra para instalar cable es sumamente elevado, y se trata de aprovechar el cable telefónico que ya existe.

U

UTP

Siglas de "Unshielded Twisted Pair". Par trenzado no-blindado. Ver Twisted-Pair.

W

WAN

Siglas de "Wide Area Network". Se llama así a la RED que se extiende sobre distancias muy grandes y que generalmente depende de líneas de comunicación para su funcionamiento correcto.

X

X.25

Estándar del CCITT que define el protocolo de comunicaciones por el que una computadora puede accesar una red de conmutación en paquetes (packet switching). En general cuando se habla de X.25 se trata de una familia de protocolos que son: X.3, X.28, etc..

X.500

Estándar de CCITT para el manejo de directorios en una red de área distribuida.

X/Open

Organización dedicada a estandarizar los niveles más altos de UNIX para poder crear un medio ambiente común de aplicaciones y mejorar la interoperabilidad de sistemas.

# #

10Base2

La implementación de Ethernet del estándar del IEEE 802.3 en cable coaxial delgado (thin coax). También se le conoce como thin Ethernet o Thin Net, corre a 10MBps. Máxima longitud por segmento 200 metros.

DEL NEGOCIO

a

Al portador

Título o valor en el que no se consigna el nombre del propietario, es pagado al tenedor, y se trasmite por simple entrega.

c

Cheque

Título de crédito expedido a cargo de una institución de crédito, por quien está autorizado por ésta el efecto, conteniendo la orden incondicional de pagar una suma de dinero a la vista al portador o a la orden de una persona.

Cheque de caja

Es el que giran las sociedades de crédito a las instituciones con cargo de sus propias dependencias o sucursales libradas, para posteriormente cargárselos al librador titular de la cuenta; éste deberá firmarlo y también el talón de la chequera de caja.

Cheque para abono en cuenta

El librador o el tenedor pueden prohibir que el cheque se pague en efectivo insertando en éste la cláusula "para abono en cuenta". En tal caso el librador tendrá que abonar el cheque en la cuenta del tenedor.

Crédito

Operación de préstamo de recursos financieros por confianza y análisis a un sujeto o empresa disponible contra una promesa de pago. Adelantar fondos o bien conceder plazo para un pago exigible; por ello, desde el punto de vista jurídico, el crédito puede considerarse como un préstamo o como una venta a plazos. En la práctica se formaliza ya sea por un movimiento de fondos o por el otorgamiento de una firma.

Crédito Personal

Es el tipo de crédito en el cual la firma del acreditado es la garantía.

Crédito en cuenta corriente

Es el contratado por el cual el acreditante se obliga a poner una suma de dinero a disposición del acreditado, la cual puede ser usada por éste una o varias veces, es decir, que tiene derecho a redisponer de las cantidades que abone en cuenta de su adeudo, antes del vencimiento de la operación.

d

Déposito a la vista

Es aquella operación, también denominada cuenta de cheques, en la cual el depositante entrega su dinero sin reservas al banco, con la certeza de que puede disponer del mismo en cualquier momento, sin intereses retirables a la vista mediante cheques.

Depósito a plazo

Depósito bancario hecho por un número convenido de días, meses o años a una tasa fija de rendimiento. Los depósitos a plazo producen una tasa de interés más alta que otros depósitos debido a que el dinero está comprometido.

i

Interés

Renta o ganancia del capital financiero que se obtiene por cobrar o pagar los depósitos bancarios.

Inversión

Se define como las erogaciones que se llevan a cabo con la esperanza de obtener posteriormente una utilidad.

l

Límite de crédito

Es el importe máximo de la deuda que se le permitirá tener a un cliente. Se fija basándose en sus necesidades probables de compra, así como en los antecedentes de su puntualidad en los pagos y en la capacidad financiera de la empresa otorgante.

o

Operaciones bancarias

Son aquellas que realizan los bancos en relación con sus funciones economico-financieras. Son variadas, pero se pueden dividir en pasivas y activas, según el banco aparezca como deudor o acreedor a consecuencia de la operación.

p

Pago

Satisfacción de una deuda en un sistema monetario. Su importancia se deriva de sus consecuencias con respecto al dinero y al crédito. Existen pagos al contado y diferidos.

Pago bancario

Es aquel que se efectúa por medio de los bancos.

Plazo

Periodo, que transcurre entre el inicio y la terminación de un contrato; término que se da para pagar o satisfacer una cosa.

Plazo fijo

Fecha de vencimiento determinada. Indica que el periodo de tiempo considerado tiene duración fija.

Préstamo

Acto de entregar u obtener dinero u otra cosa, para que por algún tiempo se tenga en uso de ello, con obligación de restituir igual cantidad o la cosa misma.

Préstamos directos

Los préstamos directos pueden considerarse como la operación clásica del crédito bancario, ya que para su otorgamiento no se exige más garantía que la que ofrece el sujeto de crédito como persona, de acuerdo con sus actividades de buena solvencia moral y económica.

S

Saldo

Es la diferencia entre el movimiento deudor y el movimiento acreedor de una cuenta.

Salvo buen cobro

Es la condición mediante la cual el banco toma documentos de cobro inmediato y remesas en camino, para abono en la cuenta de cheques de sus clientes, procediendo â abonar los documentos después de que le han sido liquidados (compensados).

t

Tarjeta de crédito

Es un instrumento de identificación que se utiliza para que a una persona, a la que un banco de depósito le ha concedido un crédito en cuenta corriente, pueda ejercerlo a la presentación de la misma hasta por el monto máximo convenido.

BIBLIOGRAFIA

LIBROS:

Business Re-Engineering with technology. Donovan John; Cambridge Technology Group, Inc.; 1993 Cambridge USA.

Client/Server Architecture. Berson A.; McGraw Hill; 1992 New York USA.

Client/Server Computing. Application Design Guidelines a Distributed Relational Data Perspective. IBM International Technical Support Centers; 1991 San Jose Cal.; USA.

Client/Server Computing Architecture, Applications, and Distributed Systems Management. Bruce Elbert, Bobby Martyna; Artech House, inc.; 1994 USA.

Client Server Computing. Commercial Strategies. Caroline Cheppell, et all; OVUM Ltd; 1991 England.

Computer Networks. Tanenbaum A.S.; Prentice Hall; 1988 USA.

Contabilidad Bancaria. Andrés Aguirre Montera; México.

Distributing Applications Across, DCE and Windows NT. Ward Rosenberry and Jim Teague; O'relly & Associates, inc.; 1993 USA.

Distributed Systems. Sape Mullender; ACM Press Frontier Series; 1991.

El Modelo Relacional y DB2. Infomedia S.A. de C.V.; México D.F.; 1992.

Fast Track to SYBASE for Netware. SYBASE, Inc.; 1993.

Implementing Production-Quality Client/Server - Systems. Bochenski Barbara; Wiley; 1994 USA.

NOVELL NetWare v3.11 TCP/IP Transport Supervisor's Guide. Novell, Inc; 1991 Utah USA.

Reengineering the Corporation. Hammer Michael & Champy James; Harper Business; 1993 USA.

Relational Database Design. Paul Winsberg; SYBASE, Inc.; 1990.

SQL Básico. Infomedia S.A. de C.V.; México D.F.; 1992.

TCP/IP Network Administration. Craig Hunt; O'Reilly & Associates, Inc.; 1993 USA.

Visual Basic for Dummies. Wallace Wang; IDG Books Worldwide, Inc.; 1994 San Mateo California.

Visual Basic Language Reference. Microsoft Corporation; 1992 USA.

Visual Basic 3.0. Ddemesis Centro Educacional; México D.F.; 1993.

REVISTAS

Aplicando Cliente/Servidor en la banca. Marco A. Maytorena; Banca Electrónica; Año 1 No. 8; julio 1994; México.

DCE: Construyendo el futuro distribuido. Michael D. Millkin; BYTE México; Año 7 No. 77; 1994; México.

Distributed Computing and the OSF/DCE. Bloomer Jhon; Dr. Dobbs Journal; No. 227; 1994; San Mateo California USA.

Panorama. Banca Electrónica; Año 1 No. 2; enero 1994; México.

Servidores de Aplicaciones. PC Magazine en Español; Vol. 5 Núm. 6; 1994.

Tendencias en Plataformas Financieras de Servicios a Clientes. Alfredo Piquer; Banca Electrónica; Año 2 No. 14; enero 1995; México.

Twelve Steps to Successful Client/Server. Shaku Atre; DBMS; Vol. 7 Núm 5; 1994; USA.

Unraveling Client/Server Arquitectures. Herb Edelstein; DBMS; Vol. 7 Núm 5, 1994, USA.

Working With Sockets. Antoniette Agracewics; UNIX Review; Vol. 12 No. 6; 1994; USA.