Cliente/Servidor

Índice

- 1. Antecedentes
- 2. Nociones generales de la arq. C/S
- 3. Características de los sistemas C/S
- 4. Estructura de una aplicación y su relación con C/S
- 5. Diseño de aplicaciones C/S
- 6. Ventajas y Desventajas de C/S
- 7. Componentes de la arquitectura C/S
- 8. Middleware
- 9. Arquitectura de 3 niveles
- 10. Evolución de C/S. P2P

Bibliografía

- R. Orfali, D.Harkey, J. Edwards

  Cliente/Servidor y objetos: Guía de Supervivencia

  3. Edición McGraw-Hill Interamericana México D.F. 2002

- G. Hamilton, R. Catell, M. Fisher

  JDBC Database Access with Java. A tutorial and Annotated Reference

  Addison Wesley 1997

Arquitectura Cliente/Servidor. Antecedentes

- Existen diversos puntos de vista sobre la manera en que debería efectuarse el procesamiento de datos, aunque la mayoría opina, que nos encontramos en medio de un proceso de evolución que se prolongará todavía por algunos años y que cambiará la forma en que obtenemos y utilizamos la información almacenada electrónicamente.
- El principal motivo detrás de esta evolución es la necesidad que tienen las organizaciones (empresas o instituciones públicas o privadas), de realizar sus operaciones más eficientemente, debido a la creciente presión competitiva a la que están sometidas, lo cual se traduce en la necesidad de que su personal sea mas productivo, que se reduzcan los costos y gastos de operación, al mismo tiempo que se generan productos y servicios más rápidamente y con mejor calidad.
- En este contexto, es necesario establecer una infraestructura de procesamiento de información, que cuente con los elementos requeridos para proveer información adecuada, exacta y oportuna en la toma de decisiones y para proporcionar un mejor servicio a los clientes y ciudadanos.

Arquitectura Cliente/Servidor. Antecedentes

- **El modelo Cliente/Servidor** reúne las características necesarias para proveer esta infraestructura, independientemente del tamaño y complejidad de las operaciones de las organizaciones públicas o privadas y, consecuentemente desempeña un papel importante en este proceso de evolución.

Arquitectura Cliente/Servidor. Antecedentes

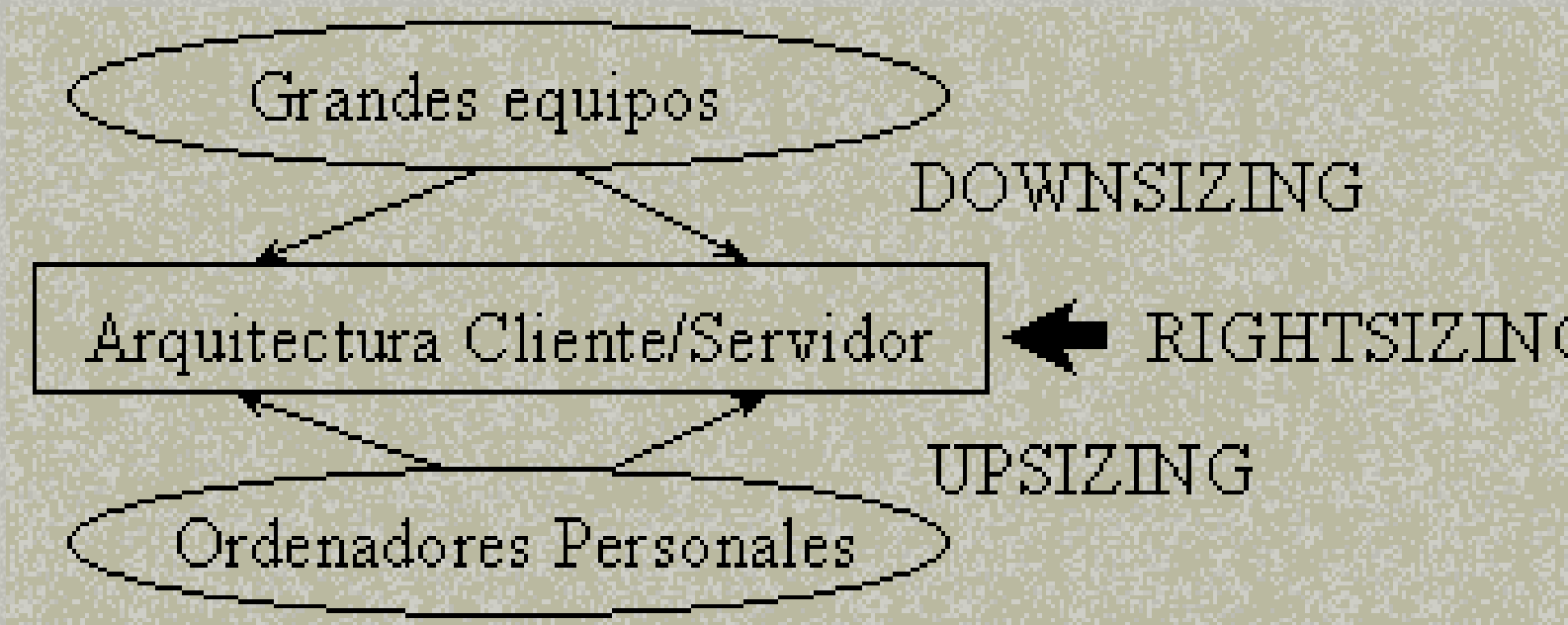
- ***Downsizing***. Es la migración de aplicaciones a plataformas de cómputo menores con la intención de obtener mayor flexibilidad, eficiencia, reducción de costos y autosuficiencia para los usuarios.

- ***Upsizing***: Es la consolidación de usuarios finales o aplicaciones y datos de redes LANs en plataformas de cómputo mayores, incrementando la facilidad de acceso, capacidad y/o rendimiento.

- ***Rightsizing***. Consiste en la selección de tecnologías de información adecuadas para la solución de la problemática de los negocios y servicios, tales como mejor respuesta al mercado, un adecuado servicio a los clientes y ciudadanos y un mayor aprovechamiento en el uso de la tecnología y de los recursos.

Arquitectura Cliente/Servidor. Antecedentes

La arquitectura Cliente/Servidor es el resultado de la integración de dos culturas. Po un lado, la del Mainframe que aporta capacidad de almacenamiento, integridad y acceso a la información y, por el otro, la del computador que aporta facilidad de uso (cultura de PC), bajo costo, presentación atractiva (aspecto lúdico) y una amplia oferta en productos y aplicaciones.



Ventajas de los “mainframes”

- Arquitectura más barata
- Gestión de grandes procesos
- “Downsizing” es sólo una moda

2. Arquitectura C/S

Conceptualmente son parte de la noción de sistemas abiertos.

- Conectar una variedad de ordenadores con diferentes hardware y software para trabajar coordinadamente con el fin de lograr los objetivos del usuario.

El objetivo de los sistemas abiertos consiste en lograr la interoperabilidad.

- Estado que caracteriza a múltiples sistemas heterogéneos que se comunican y contribuyen a la terminación de una tarea común.

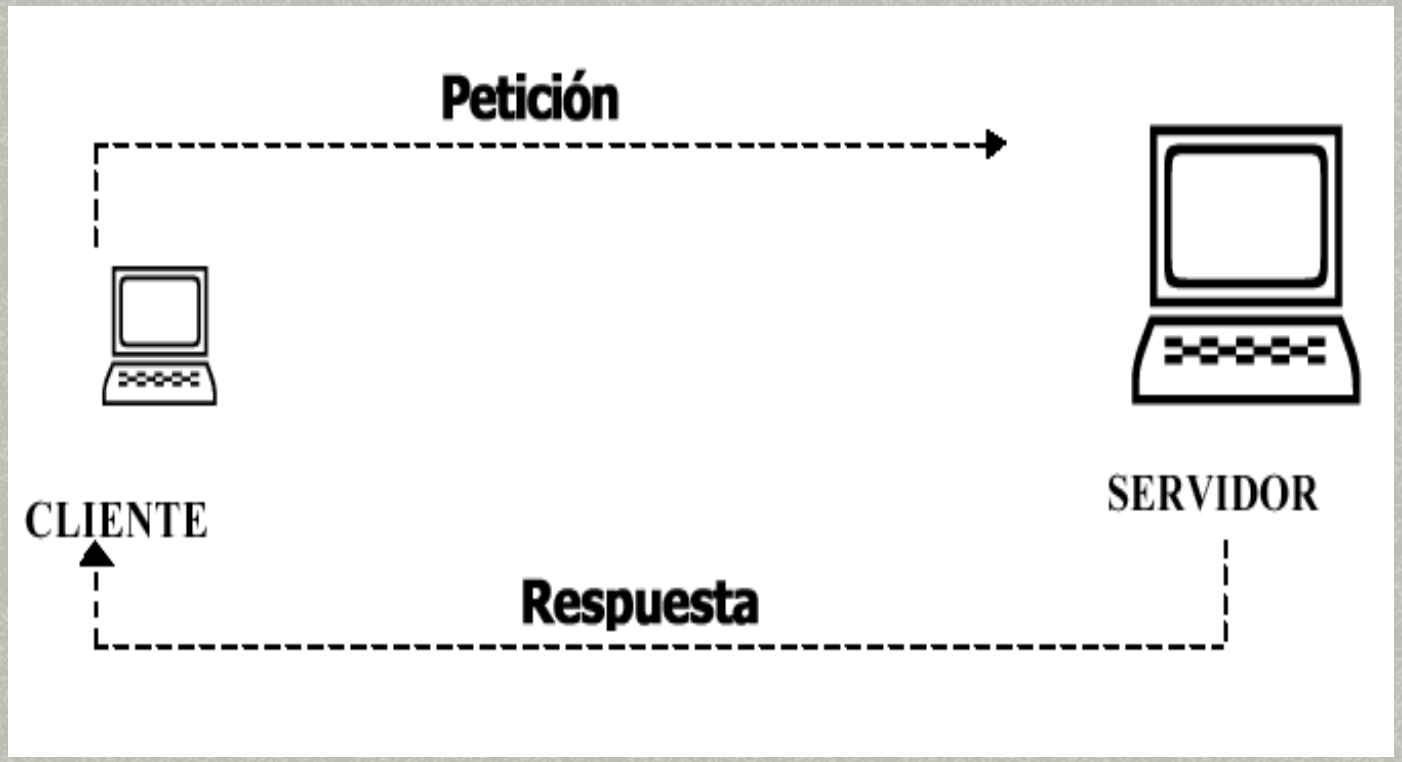
2. Filosofía Cliente/Servidor

Desde un punto de vista conceptual:

- «Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información»

- Cliente (frontend): consumidor de servicios.
- Servidor (backend): proveedor de servicios

2. Arquitectura Cliente/Servidor (2 niveles)



2. Arquitectura Cliente/Servidor

Cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

2. ¿Qué es Cliente/Servidor?

En términos de arquitectura:

Los distintos aspectos que caracterizan a una aplicación (proceso, almacenamiento, control y operaciones de entrada y salida de datos) en el sentido más amplio, están situados en más de un computador, los cuales se encuentran interconectados mediante una red de comunicaciones».

Es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información; estableciendo así un enlace de comunicación transparente entre los elementos que conforman la estructura.

2. ¿Qué es Cliente/Servidor?

- ***IBM define al modelo Cliente/Servidor***

- «Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores".

2. ¿Qué es C/S: requisitos básicos?

- Cliente y servidor pueden estar en dos ordenadores distintos
- Cliente y servidor no son aplicaciones completas → se complementan
- Cada subsistema de la aplicación en el sitio más adecuado
- Varios clientes → replicado del servidor

2. Arquitectura C/S

- La potencia descansa en el concepto de División de Funciones.

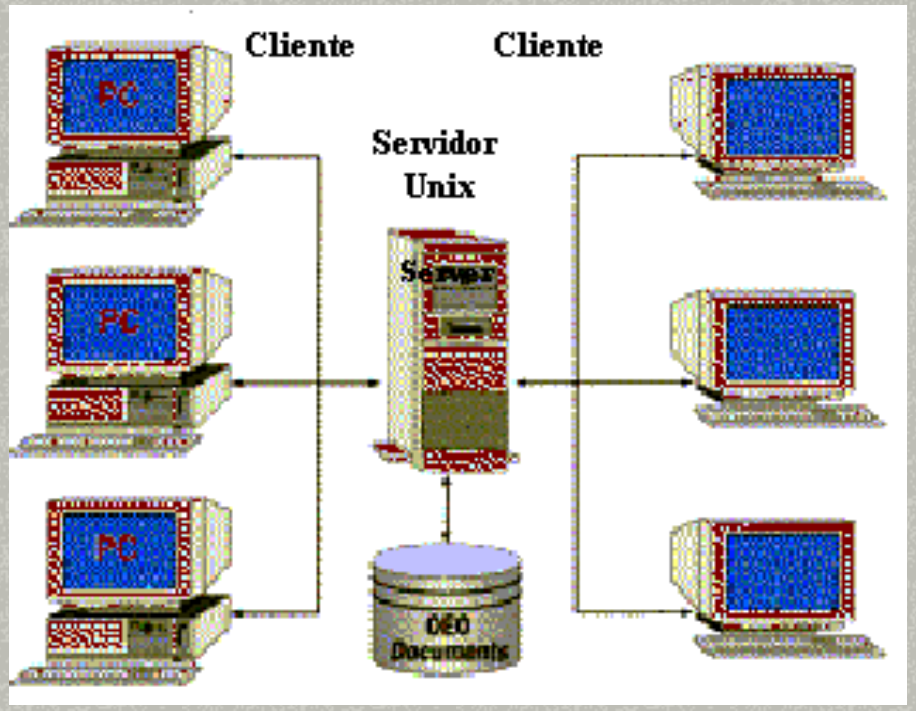
  Cliente. Ofrece un GUI y ejecuta programas de interés para el usuario.

  Servidor. Gestiona el acceso a los datos y realiza funciones de control y seguridad.

Podemos tener ***lo mejor de ambos mundos***.
Fuerza que se cumpla el principio de Modularidad.

2. Arquitectura C/S

- La arquitectura Cliente/Servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.



3. Características de los sistemas Cliente/Servidor

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

3. Características de los sistemas Cliente/Servidor

- **Servicio**: se trata principalmente de una relación entre procesos en máquinas diferentes. El proceso servidor proporciona servicios. El cliente consume servicios. Este mecanismo proporciona una separación clara de funciones basada en la idea de servicio

- **Recursos Compartidos**. Un servidor puede atender a muchos clientes al mismo tiempo y regular sus accesos a los recursos compartidos

3. Características de los sistemas Cliente/Servidor

- **Protocolos Asimétricos**: Hay una relación 1 a n entre el servidor y los clientes. Los clientes siempre inician el diálogo pidiendo servicios. Los servidores esperan pasivamente las peticiones de los clientes.

- **Transparencia de la Localización**: El servidor es un proceso que puede residir en la misma máquina que los clientes o en una máquina diferente en una red. El software Cliente/Servidor oculta la localización del servidor a los clientes, redirigiendo las llamadas a los servicios cuando sea necesario.

3. Características de los sistemas Cliente/Servidor

- **Independencia**: El software cliente/servidor ideal es independiente del hardware y del sistema operativo.

- **Intercambios basados en Mensajes**: Los clientes y servidores interactúan utilizando un mecanismo de paso de mensajes. El mensaje es el soporte para las peticiones de servicios y las respuestas

3. Características de los sistemas Cliente/Servidor

- **Encapsulación de Servicios**: El servidor es un especialista. Un mensaje le dice al servidor que servicio se solicita; es tarea del servidor determinar como realizar el trabajo. Los servidores se pueden actualizar sin afectar a los clientes siempre y cuando no se modifique su interfaz.

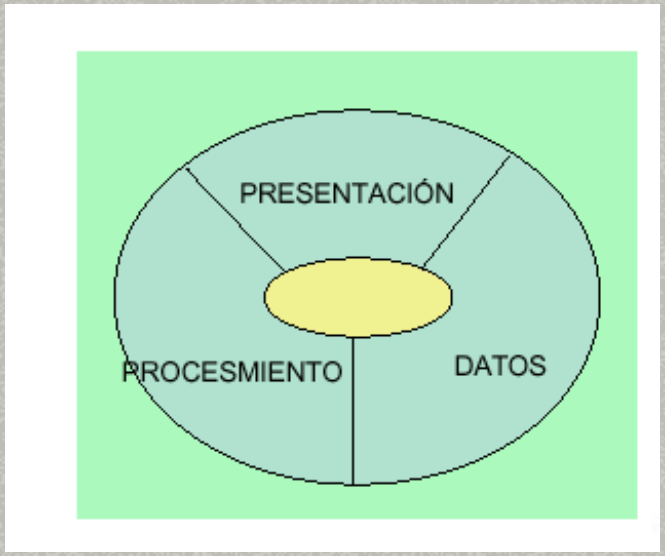
- **Escalabilidad**: Los sistemas cliente/servidor se pueden escalar horizontal y verticalmente. Horizontalmente significa añadir o eliminar clientes. Verticalmente significa migrar a máquinas servidoras más potentes o multiservidores.

3. Características de los sistemas Cliente/Servidor

- **Integridad**: El código y los datos del servidor se mantienen centralizados, con lo que es menos costoso su mantenimiento y controlar la integridad de los datos compartidos.

4. La Estructura de una Aplicación y su Relación con Cliente/Servidor

- Toda aplicación de software tiene tres funciones fundamentales: administración de los datos, lógica de la aplicación (procesos) y lógica de la presentación (interfaz de usuario).



4. La Estructura de una Aplicación y su Relación con Cliente/Servidor

Existe un conjunto de variantes de la arquitectura Cliente/Servidor, dependiendo dónde se ejecutan las diversas funciones de una aplicación (qué asume el cliente y qué el servidor)

- **Presentación Distribuida.** En este modelo, se distribuye la presentación entre el cliente y el servidor, pero éste último ejecuta todos los procesos y almacena la totalidad de los datos.
- **Presentación Remota.** Aquí, la interfaz del usuario está completamente en el cliente, la presentación soporta la captura de datos, incluyendo una validación parcial de los mismos y una presentación de las consultas. La lógica de la aplicación y los datos está en el servidor.

4. La Estructura de una Aplicación y su Relación con Cliente/Servidor

- **Proceso Distribuido.** Se da cuando la presentación está en el cliente, la base de datos está en el servidor y la lógica de la aplicación está distribuida entre el cliente y el servidor.
- **Gestión de Datos Remota.** Para este modelo, tanto la presentación como los procesos de la aplicación residen en el cliente, mientras que las bases de datos permanecen centralizadas en el servidor.
- **Bases de Datos Distribuidas.** Este último modelo, la presentación, los procesos de la aplicación y parte de los datos de la Base de Datos están en el cliente; el resto de los datos están en el servidor.

4. La Estructura de una Aplicación y su Relación con Cliente/Servidor. Modelos Básicos

- **Servidores amplio**s, cuando la parte preponderante de la aplicación está en el servidor (presentación distribuida). En este modelo, el cliente aporta la interfaz gráfica de usuario e interactúa con el servidor a través de llamadas a procedimientos remotos (RPC).

- **Clientes grande**s, cuando la mayor parte de la aplicación corre en el cliente (gestión de datos remota y bases de datos distribuidas); aquí se aprovecha la potencia de procesamiento del cliente y se minimiza el costo del procesamiento en el servidor. Este modelo se emplea en software de apoyo de decisiones y personal. Los servidores de bases de datos y archivos son ejemplos de clientes grandes.

4. La Estructura de una Aplicación y su Relación con Cliente/Servidor. Arquitecturas

- Arquitectura de dos niveles.
  - La lógica de la presentación está integrada ya sea a la interfaz del usuario en el cliente o a la base de datos en el servidor (o a ambas); ejemplo: servidores de archivos y de bases de datos.

- Arquitectura de tres niveles.
  - El proceso de la aplicación ocupa el plano intermedio; está separada tanto de los datos como de la presentación. Los procesos se pueden administrar y desplegar en forma autónoma, sin relación con la interfaz gráfica del usuario y la base de datos.

5. DISEÑO

- No se cuentan con metodologías claras para el diseño
- Las metodologías tradicionales de desarrollo de sistemas de información se quedan cortas (ej. Modelo E/R, orientado a objetos)

5. Diseño Etapas

- 5.1 Planificación y Análisis de Sistemas
- 5.2 Costos
- 5.3 DISEÑO
- 5.4 Desarrollo
- 5.5 Aplicaciones de Prueba
- 5.6 Entrega del sistema

5.1 Planificación y Análisis de Sistemas

- En esta etapa se debe:
  - Definir los objetivos y las metas
  - Indicar el personal que esté involucrado en el proyecto
  - Especificar la justificación de migración hacia el nuevo sistema
  - Presentar los beneficios que se pretender obtener

5.1 Planificación y Análisis de Sistemas

- Hay que realizar un análisis completo sobre la problemática existente en la organización:
  - Necesidades de la empresa y volúmenes de la información que maneja
  - Restricciones de accesos a la información real para los usuarios de los distintos niveles empresariales
  - Aplicaciones críticas para la empresa
  - Frecuencia de transacciones diarias
  - Identificar las funciones del hardware
  - Dificultades en la comunicación y manejo de documentos entre grupos de trabajo
  - Visión del crecimiento (usuarios, aplicaciones)
  - Sistemas de red actual y sus alternativas de crecimiento.

5.2 Costos

- El estudio debe reflejar cual sería el impacto para la empresa, el migrar hacia el nuevo sistema en términos de corto, medio y largo plazo.

  ¿Cuándo se recuperará la inversión de las adquisiciones?

  Costos cuantitativos, cualitativos y periodos de tiempo.

5.3 DISEÑO

- Se recomienda centrarse primero en la estructura lógica del sistema antes de involucrarse en los detalles físicos. Se deben seguir los siguientes pasos:
  - Asignar los requerimientos funcionales entre clientes y servidores.
  - Distribuir los recursos entre los servidores

5.3 DISEÑO. Distribuir los recursos entre los servidores

- Aspecto lógico
  - Seleccionar arquitectura de 2 o 3 niveles
  - Mencionar el tipo y las características del software que se piensa adquirir para incorporarlo con el ya existente
- Aspecto técnico
  - Características de los componentes
  - Topología de la red

Para el diseño de aplicaciones C/S se proponen unos heurísticos

5. Reglas heurísticas para el diseño de aplicaciones Cliente/Servidor

- Disponer del mayor procesamiento posible en el cliente
- Administrar todos los recursos compartidos en el servidor
- Minimizar los datos transferidos entre clientes y servidores
- Evitar la centralización de servicios
- .....

5.4 Desarrollo

- Se efectuarán las gestiones respectivas a la adquisición de los componentes físicos y/o software que se requieran
- Realizar la instalación e interconexión real, integrando y distribuyendo físicamente todos los recursos dentro de la red.

5.5 Aplicaciones de Prueba

- Principales elementos a verificar:
  - Interfaz gráfica de usuario
  - Ayuda de contexto y otra documentación en línea
  - Manejo de errores
  - Integración
  - Impacto al entorno de operación
  - Flujo de información
  - Ejecución
  - Administración y operación
  - Seguridad

5.6 Entrega del sistema

- "*NO es necesaria*" la etapa de formación

6. Ventajas del modelo C/S

- Mejora el servicio prestado a los clientes
- Descarga de trabajo a los ¨mainframes¨
- Aumenta la productividad
- Mejora la compartición e integridad de los datos. Disminuye costes de operación
- Reduce el tráfico en la red
- Reduce el tiempo de desarrollo: reutilización, portabilidad, modularidad.

Ventajas para las organizaciones

- Se reducen los costos de producción de software y se disminuyen los tiempos requeridos. Esto es así, pues, para la construcción de una nueva aplicación pueden usarse los servidores que estén disponibles, reduciéndose el desarrollo a la elaboración de los procesos del cliente, según los requerimientos deseados. Lo anterior disminuye los costos internos del área de sistemas. Además, se pueden obtener ventajas importantes al reducir el costo del hardware requerido, llevando las aplicaciones a plataformas más baratas, aprovechando el poder de cómputo de los diferentes elementos de la red, y facilitando la interacción entre las distintas aplicaciones de la organización.

- El esquema Cliente/Servidor también contribuye a una disminución de los costos de formación de personal, pues favorecen la construcción de interfaces gráficas interactivas, las cuales son más intuitivas y fáciles de usar por el usuario final.

Ventajas para las organizaciones

- Facilita el suministro de información a los usuarios. Esto es así, porqu por un lado proporciona una mayor consistencia a la información de la organización, al contar con un control centralizado de los elementos compartidos, y por otro, porque facilita la construcción de interfaces gráficas interactivas, las cuales pueden hacer que los "datos" se conviertan en "información".

- Permite llevar más fácilmente la información a donde se necesita, contribuye a aumentar su precisión pues se puede obtener de la fuent (el servidor) y no de una copia en papel o en medio magnético.

- La habilidad de integrar sistemas heterogéneos es inherente al model Cliente/Servidor, pues los clientes y los servidores pueden existir en múltiples plataformas y tener acceso a datos de cualquier sitio de la red. Un cliente puede integrar datos de diferentes sitios para presentarlos, a su manera, al usuario final.

Ventajas para las Organizaciones

- Al favorecer la construcción de interfaces gráficas interactivas y el acceso transparente a diferentes nodos de la red, se facilita el uso de las aplicaciones por parte de los usuarios, lo cual aumenta su productividad.
- Favorece la adaptación a cambios en la tecnología, pues facilita la migración de las aplicaciones a otras plataformas y, al aislar claramente las diferentes funciones de una aplicación, hace más fácil incorporar nuevas tecnologías en ésta.
- Hoy en día tienen mucha importancia los conceptos de sistemas abiertos e interoperabilidad, los cuales están íntimamente ligados con el concepto de Cliente/Servidor.

Ventajas para las Organizaciones

- Hace algunos años cuando una organización decidía comprar un equipo, no podía evitar quedar casada con la compañía vendedora, pues ésta era la única que podía prestar servicios de mantenimiento y actualización. Dado que los equipos de diferentes vendedores no tenían nada en común, cualquier desarrollo posterior a la primera compra implicaba compras al mismo vendedor, por factores de compatibilidad. Por esta razón se reducía la competencia, pues las grandes compañías acaparaban el mercado y los clientes o ciudadanos no podían cambiar de proveedor.
  Con este panorama surgió la idea de la implantación de estándares, porque ellos posibilitan el intercambio de información de manera coherente entre productos de diferentes vendedores. Esto permite a nuevos proveedores la oportunidad de entrar al mercado y a los clientes, la oportunidad de cambiar de proveedor.

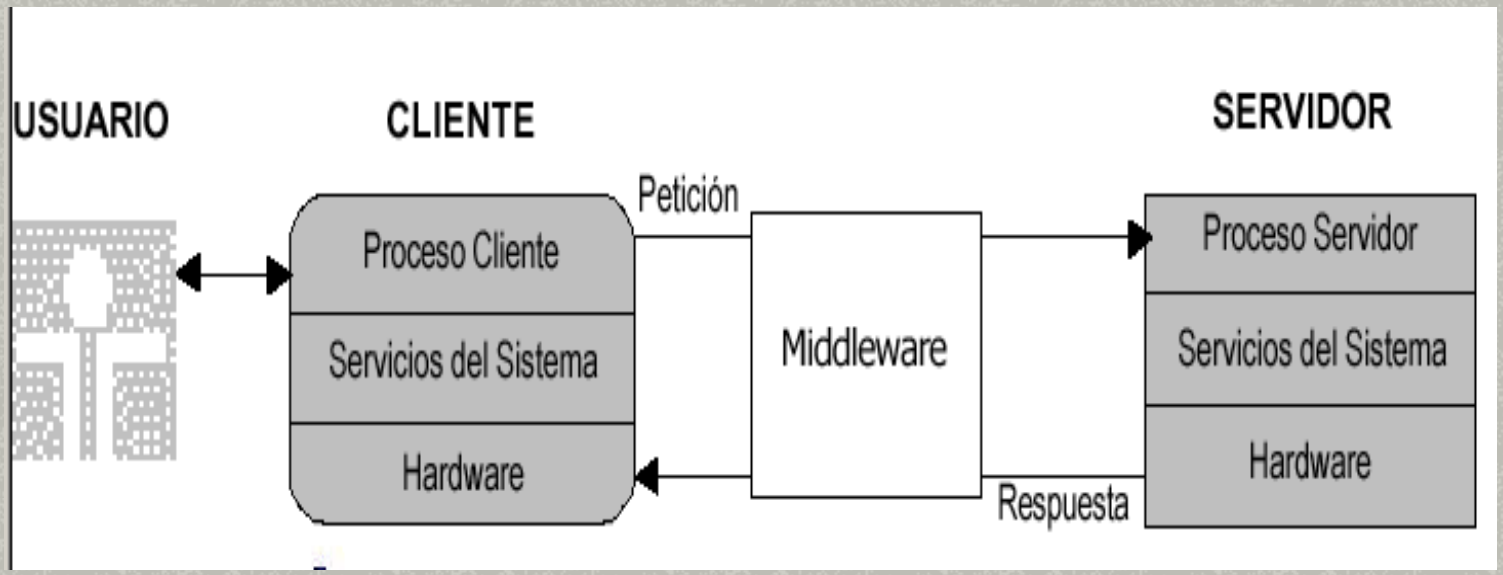
6. Desventajas del modelo Cliente/Servidor

- Complejidad
- Falta de estándares. Necesidad de trabajar con diferentes productos
- Uso intensivo de los recursos en la parte cliente
- Acceso distribuido
- Congestión de la red
- Difícil asegurar un grado elevado de seguridad

7. Componentes de la arquitectura Cliente/Servidor

- Cliente
- Servidor
- Middleware. Infraestructura de comunicaciones

7. Componentes



7. Clientes

- Entidad por medio de la cual un usuario solicita un servicio, realiza una petición o demanda el uso de recursos. Este elemento se encarga, básicamente, de la presentación de los datos y/o información al usuario en un ambiente gráfico.
- Soportan el código de aplicación no relacionado directamente con los datos. El código se genera utilizando herramientas de desarrollo de aplicaciones. Implementan los diálogos interactivos con los usuarios, los tratamientos de los mensajes y la presentación de resultados.
- Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

7. Funciones del cliente

- Administrar la interfaz de usuario
- Aceptar datos usuario.
- Procesar la lógica de la aplicación
- Generar las solicitudes para la BD
- Transmitir las solicitudes de la BD al servidor
- Recibir los resultados del servidor.
- Dar formato a los resultados.
- Captura y validación de los datos de entrada

7. Servidores

- El servidor es la entidad física que provee un servicio y devuelve resultados; ejecuta el procesamiento de datos, aplicaciones y manejo de la información o recursos.
- Aseguran el almacenamiento, distribución, gestión de la disponibilidad y de la seguridad de los datos. Permiten el acceso a los datos.
- El proceso del servidor es reactivo, es decir, realiza una función posterior a una petición o la ejecución de una transacción requerida por el cliente, o bien por otro servidor.

7. Funciones del servidor (SQL)

- Aceptar las solicitudes sobre la BD de los clientes.
- Procesar las solicitudes sobre la BD
- Dar formato a los resultados y transmitirlos al cliente.
- Llevar a cabo la verificación de integridad.
- Mantener los datos generales de la BD.
- Proporcionar control de acceso concurrente.
- Llevar a cabo la recuperación.
- Optimizar el procesamiento de consultas/actualización.

7. Diferentes tipos de servidores

- A distintos sistemas con diferentes arquitecturas se les ha denominado Cliente/Servidor. Sin embargo se clasifican basándose en su funcionalidad
  - Servidores de Ficheros
  - Servidores de Bases de Datos
  - Servidores de Transacciones
  - Servidores de Objetos
  - Servidores de Web
  - .....

7. Diferentes tipos de servidores

- **Servidores de Ficheros.**
  - El cliente formula peticiones de registros, a través de una red, al servidor de ficheros
- **Servidores de Bases de Datos**
  - El cliente formula peticiones SQL al servidor de BD. De interés para los casos en los que se requieren preguntas ad-hoc

7. Diferentes tipos de servidores

- **Servidores de Transacciones**
  - El cliente invoca a procedimientos que residen en el servidor. Dichos procedimientos ejecutan un conjunto de sentencias SQL. (OLTP)

- **Servidores de Objetos**
  - Los objetos clientes se comunican con los objetos servidores usando ORB. El cliente invoca un método de un objeto remoto. El ORB localiza una instancia de esa clase de servidor de objetos, invoca el método solicitado y envía los resultados a los objetos cliente.

7. Diferentes tipos de servidores

- **Servidores de Web**

  Se usan para comunicación entre empresas a través de Internet.

7. Diferentes tipos de servidores

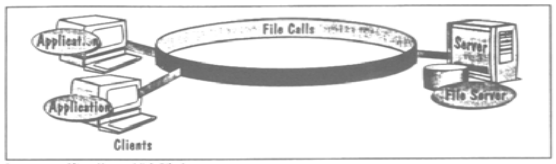


Figure 2-2. Client/Server With File Servers.

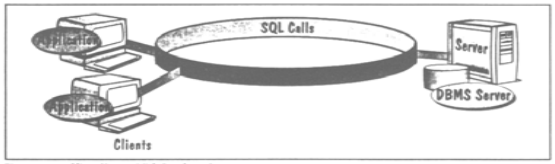


Figure 2-3. Client/Server With Database Servers.

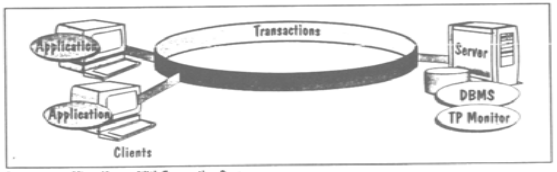


Figure 2-4. Client/Server With Transaction Servers.

7. Diferentes tipos de servidores

Figure 2-5. Client/Server With Groupware Servers.

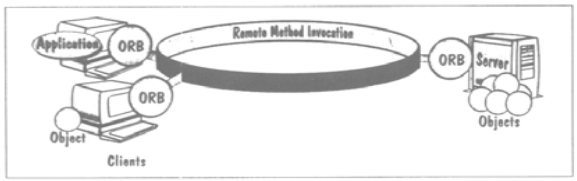


Figure 2-6. Client/Server With Distributed Objects.

8. Middleware . Definiciones

- Software que se encuentra en el medio (MIDDLE) del sistema C/S y se ejecuta en ambos lados.
- El middleware es un módulo intermedio que no pertenece a los dominios del servidor ni a la interfaz de usuario ni a la lógica de la aplicación en los dominios del cliente. Tampoco debe confundirse con la red física en sí. El middleware es una interfaz lógica estándar de los servicios de red.

8. Middleware . Definiciones

- Software que permite a las aplicaciones cliente (de cualquier tipo) comunicarse con servidores distantes (heterogéneos) para cualquier tipo de heterogeneidad de los recursos utilizados.
- Software distribuido necesario para el soporte de interacciones entre clientes y servidores.

8 Middleware

- Se distinguen 4 categorías:
  - Transport Stack
  - NOS (Network Operating System)
  - DSM (Distributed System Management)
  - Service Specific (middleware específico de servicios)

8 Middleware

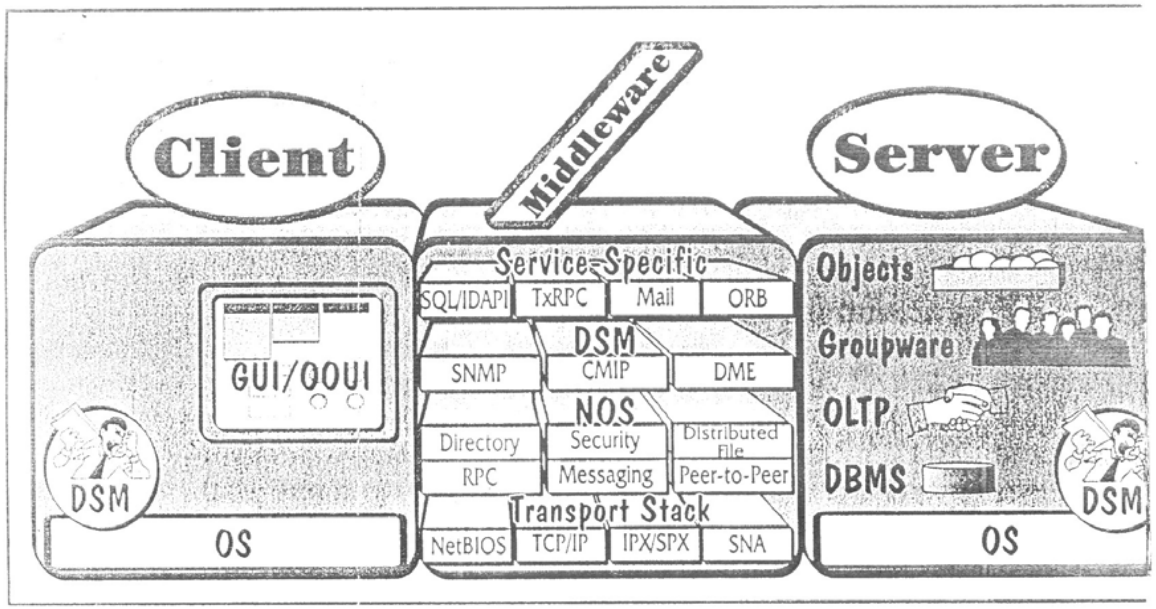


Figure 4-1. The Client/Server Software Infrastructure.

8 Middleware. Transport Stack

- Consiste en los protocolos de transporte que proporcionan comunicaciones punto a punto fiables en Redes WAN y LAN
  - NetBIOS→ Redes Microsoft
  - TCP/IP → Internet
  - IPX/SPX → Netware
  - SNA → Redes de IBM

  Por encima de este nivel se pueden tener APIs independientes, por ejemplo Sockets.

8 Middleware. NOS (Network Operating System)

- Las funciones realizadas por este middleware son:
  - 1. Extender el alcance del SO local para incluir dispositivos en red como impresoras etc.
  - 2. Proporcionar los fundamentos que ayuden a crear un sistema único de todos los recursos diversos distribuidos en la red. Ej. servicios de directorio para encontrar cosas en la red, servicios de seguridad.

8 Middleware. NOS (Network Operating System)

- Soportar la coordinación de aplicaciones que se repartan en nodos cliente/servidor.
  - Existen dos tipos de intercambios cliente/servidor:
    - Interacciones de petición/respuesta fuertemente acopladas.RPC (Remote Procedure Calls) proporciona estas interacciones.
    - Interacciones básadas en colas débilmente acopladas.Se usa MOM (Message-Oriented Midleware) para estas interacciones

8 Middleware. DSM (Distributed System Management)

- Permite que las estaciones hablen con los servicios a gestionar
- Se ejecuta en todos los nodos de la red
- Estándares actuales
  - SNMP (Simple Network Management Protocol). Para Internet. El más extendido
  - CMIP (Complex Management Information protocol)
  - DME (Distributed Management Environment). Pretende dar una solución total para sistemas de gestión de redes en entornos heterogéneos de varios fabricantes.

8 Middleware. Service Specific

- SQL/IDAPI (DBMS): permite invocar servicios basados en SQL sobre varias bases de datos de distintos fabricantes. Ej. ODBC, IDAPI (IBM…)
- TxRPC. (OLTP): permite a los clientes invocar servicios sobre varios servidores de transacciones.
- Mail (Groupware): permiten que los clientes invoquen servicios en un servidor groupware (workflow, correo electrónico etc.)
- ORB (Objects): permiten a los clientes invocar métodos que residen en servidores remotos.

8. Middleware

- Software construido sobre el protocolo de transporte con el fin de permitir el intercambio de peticiones y respuestas entre el cliente y el servidor de manera transparente.

- En resumen, trata de asegurar la transparencia respecto a las redes, los SGBD y en cierta medida a los lenguajes de acceso.

8. Transparencia a las redes

- Se deben soportar todo tipo de redes. El middleware se construirá sobre los niveles OSI y ocultará la heterogeneidad de las redes y protocolos de transporte que se usen ofreciendo una interfaz estándar de dialogo a la aplicación.

8. Transparencia a los servidores

- El middleware debe ocultar la diversidad y uniformizar el lenguaje SQL apoyandose en lo posible en los estándares.

8. Transparencia a los lenguajes

- El middleware
  - debe resolver los problemas asociados al modo diferente de trabajar, que pueden surgir entre un lenguaje, a menudo secuencial y el servidor conjuntista.
  - Debe permitir la integración de funciones de conexión a los servidores, de emisión de peticiones y recepción de respuestas para cualquier lenguaje de desarrollo utilizado por el cliente.

8. Funciones del Middleware

- **Independizar las dos entidades**: El cliente y el servidor no necesitan saber comunicarse entre ellos, sino cómo comunicarse con el módulo de middleware.
- **Traducir la información de una aplicación y pasarla a la otra.** Acepta consultas y datos recuperándolos de la aplicación cliente, los transmite y envía la respuesta de regreso. También genera los códigos de error.
- **Controlar las comunicaciones.** Da a la red las características adecuadas de desempeño, confiabilidad, transparencia y administración.

8. Funciones del Middleware

- Procedimiento de Conexión.

  Operación que permite abrir un camino desde un cliente al servidor designado por un nombre, con verificación del nombre de usuario y palabra reservada. Es necesario identificar también el nombre de la BD.

- Preparación de la petición

  Operación que permite enviar una petición con parámetros no instanciados a un servidor con el fin de preparar su ejecución.

8. Funciones del Middleware

- Ejecución de la petición

  Operación que permite enviar una orden de ejecución de con los parámetros instanciados

- Recuperación de los resultados

  Operación que permite traer todo o parte del resultado al cliente.

- Procedimiento de Desconexión

  Operación que permite cerrar el camino abierto desde el cliente al servidor.

8. Técnicas del Middleware

- Cacheo de Resultados

  Técnica que permite transferir los resultados por bloques y conservarlos sobre el cliente o servidor a fin de re-utilizarlos para responder a peticiones.

- Cacheo de Peticiones

  Conjuntos de instrucciones y lógica de procedimientos de SQL compilado, verificado y almacenado en la Base de Datos del servidor. El cliente invoca un procedimiento remoto y le transmite los parámetros requeridos a un procedimiento almacenado. Problema NO son estándar

8. Tipos de Midlleware

- El **Middleware general** es el sustrato de la mayoría de las interacciones de Cliente/Servidor. Incluye las pilas de comunicación etc.

- El **Middleware de servicios específicos** es necesario para cumplir un tipo particular de servicio de Cliente/Servidor; así, existe un middleware específico para los servidores dedicados: Middleware para bases de datos,

8. Objetivos del Middleware

- Transporte de peticiones y respuestas.
- Simplificación de la visión de usuario.
- Armonización de tipos de datos.
- Rendimiento.
- Fiabilidad.

8. Software del Middleware

- Los paquetes software que constituyen el middleware representan cuatro capas funcionales (que pueden estar imbricadas)
  - protocolo de comunicación entre un proceso cliente y otro servidor (RPC, CORBA).
  - Acceso a servidores SQL heterogéneos .
  - Control distribuido de transacciones.
  - Herramientas de desarrollo C/S.

8. Acceso a servidores SQL heterogéneos

- Situación: los SGBD proporcionan pasarelas de acceso remoto a su sistema y pasarelas para otro tipo de sistemas.
- Problema. Necesidad de proporcionar muchas pasarelas.
- Solución: En 1988, 44 vendedores de SGBD crean un consorcio denominado SQL Access Group (SAG) con el fin de proporcionar un estándar para el acceso remoto de BD.

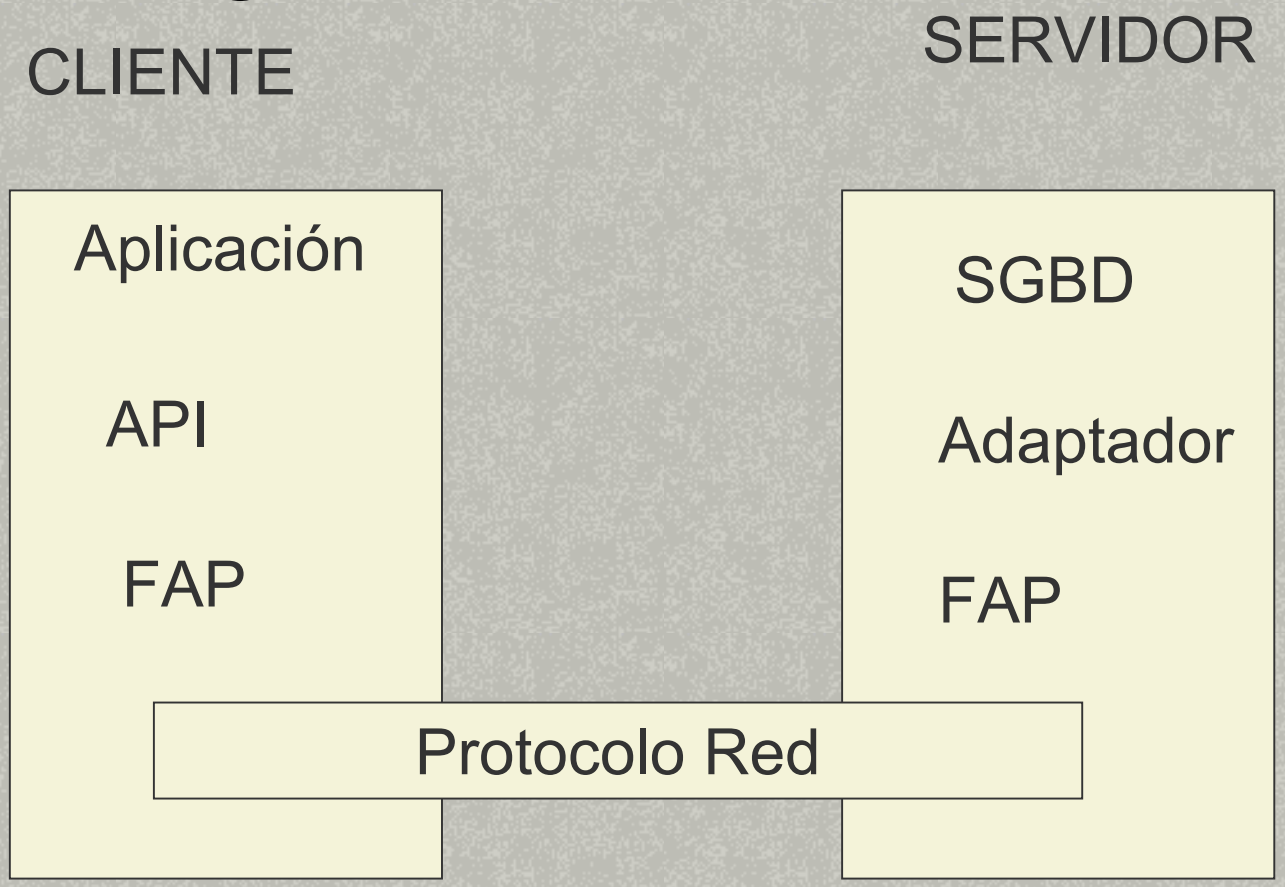
8. Acceso a servidores SQL heterogéneos

- SAG enfocó sus esfuerzos en dos direcciones:
  - SQL Call Level Interface (CLI) que define un API común para un conjunto de vendedores.
  - Interoperabilidad entre clientes y servidores utilizando formatos de mensajes y protocolos comunes. (FAP estándar)

8. Elementos importantes del middleware SQL

- **La API de SQL del cliente.** Es la parte del cliente por medio de la cual se invoca un servicio, y comprende la transmisión de la solicitud por la red y la respuesta resultante. Pero, no incluye el software que presta el servicio como tal.
- **El controlador de SQL.** Es un pequeño elemento que en tiempo de ejecución del cliente acepta las llamadas de la API, da formato a un mensaje de SQL y maneja los intercambios con el servidor.
- **Soporte de FAP (Formats and Protocols) para pilas de protocolos.** La mayoría de los proveedores soportan múltiples pilas de protocolos
- **Gateways a Base de Datos de otros proveedores.(**hacen que las BD de otros proveedores se vuelvan semejantes a las suyas**)**
- **Herramientas de administración de bases de datos. (**mediante GUI desde una estación de trabajo remota**)**
- **Herramientas frontales de desarrollo de aplicaciones gráficas y consultas.**

Acceso a servidores SQL heterogéneos



8 Tipos de APIs

- API Application Program Interface
- Tipos de APIs
  - Propietarios Ej. OCI (Oracle Call Interface)
  - Interoperables
    - CLI (Call Level Interface)
    - ODBC (Open Data Base Connectivity)

8. Funciones Principales del CLI

- CONNECT. Apertura de una conexión a un servidor.
- PREPARE. Preparación de una orden para su ejecución posterior.
- EXECUTE. Ejecución de una orden preparada.
- EXECDIRECT. Ejecución directa de una orden.
- SETCURSORNAME. Posicionamiento de un cursor.
- DESCRIBEATT. Descriptor de un atributo.
- FETCH. Lectura de tuplas siguiendo el resultado.
- ROWCOUNT. Nº de tuplas afectadas por una orden SQL
- TRANSACT. Validar o anular una transacción.
- CANCEL. Anular una orden SQL en curso de ejecución.
- .....

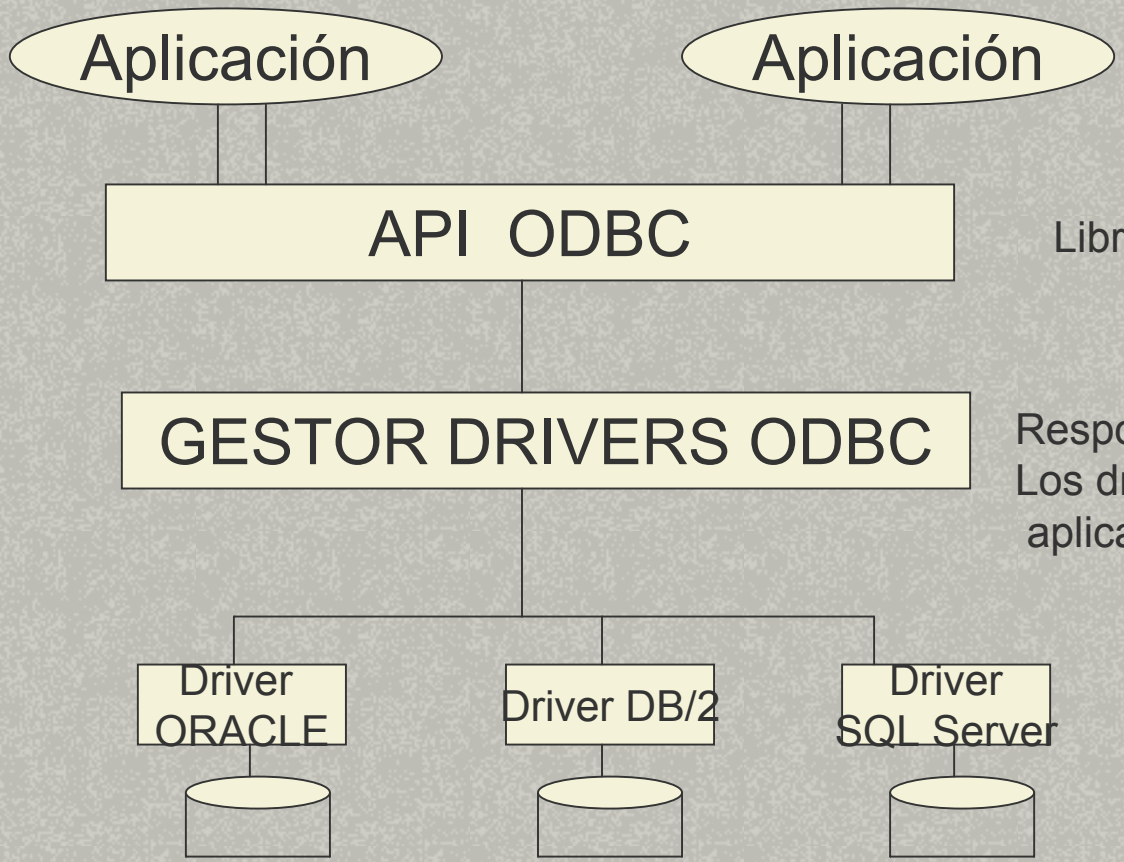
8. ODBC

- Windows API (Microsoft)
- Versión extendida del SAG CLI
- Vendedores de SGBD (IBM,Informix, Ingres..) proporcionan **Drivers** ODBC para sus servidores SQL.
- Limitación: Especificación controlada por Microsoft. Siempre es más lento que trabajando con APIS nativas.

***Driver**: Acepta una llamada CLI y la traduce al lenguaje del servidor correspondiente.Escrito para un servidor específico. Oculta las diferencias no solo respecto al SGBD sino también respecto al SO y al protocolo de red usado.*

Driver ODBC: programa que interactúa con un SGBD concreto y ofrece un API según los dictados ODBC

ODBC (Open Data Base Connectivity)



8. JDBC

- API de JAVA para acceder a una B.D. usando SQL. API de bajo nivel (se usara llamando directamente a sentencias SQL). Define cómo una aplicación JAVA puede comunicarse con una BD.
- Clases y métodos para:
  - establecer una conexión con una BD
  - enviar sentencias SQL a dicha BD
  - procesar los resultados
- Los desarrolladores de SGBD proporcionan drivers JDBC
- Código no cambia si cambiamos de SGBD, excepto la referencia al driver.

8. Módulos de JDBC

- API JDBC interfaz que accede a las funciones que se pueden realizar con el JDBC
- Gestor de drivers JDBC encargado de realizar la llamada al driver correspondiente dependiendo de lo que se haya programado a través de la aplicación JAVA.
- Driver JDBC encargado de realizar la traducción de la llamada que se ha hecho a una sentencia SQL que entienda el SGBD.

JDBC se compone de clases incluidas dentro del paquete java.sql*

8. Standard JDBC

- Basado en el X/Open SQL Call Level Interface (CLI) para interacciones cliente/servidor
- Basado en el standard ANSI SQL-92
- Adoptado por la mayoría de los vendedores de SGBD.

8. ¿Por qué no utilizar ODBC?

- ODBC está escrito en C por lo tanto los programas escritos en JAVA deberían hacer llamadas a código en C y esto crea problemas
- Traducción directa de ODBC a JDBC es difícil. (ODBC hace un uso intensivo de punteros mientras que JDBC es una interfaz orientada al objeto)
- Al estar escrito JDBC en JAVA permite su exportabilidad

8. Tipos de drivers

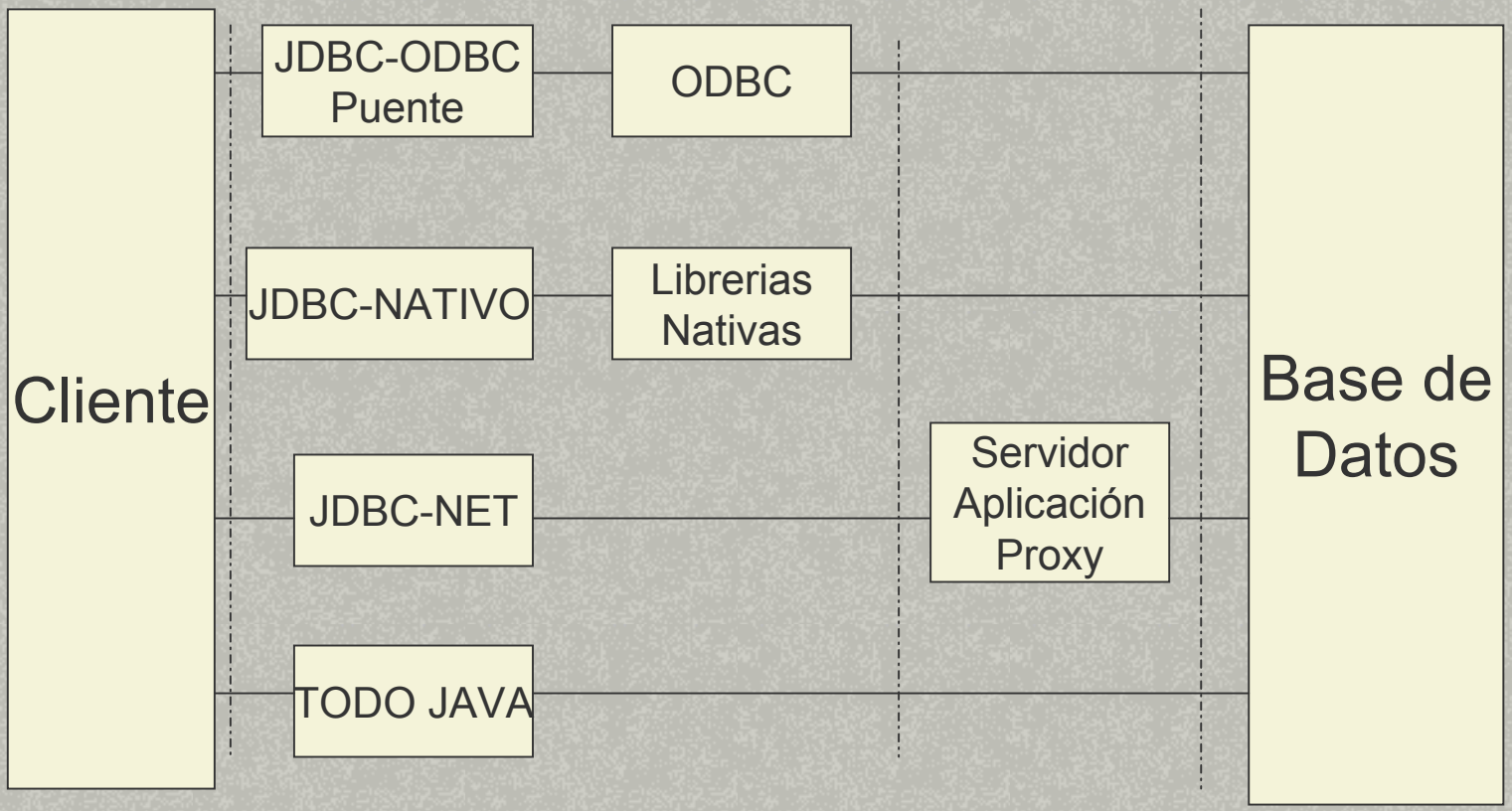
- **Puente JDBC-ODBC**. El puente se encarga de traducir las llamadas que se hacen a llamadas ODBC y de pasarlas al driver ODBC apropiado.

- **Native-API**. Convierte las llamadas JDBC a llamadas del API del SGBD. Contiene llamadas a métodos nativos en C o C++ proporcionados por los vendedores de los SGBD.

8. Tipos de drivers

- **Network-Protocol**. Traduce las funciones JDBC a un protocolo de comunicaciones independiente del SGBD, el cual traduce dichas funciones al API del SGBD específico. (tres capas). Utiliza sockets para conectarse con un software middleware que es el que trabaja con la BD.
- **Native-Protocol**. Traducen directamente las llamadas JDBC al protocolo de red que usa la BD. Se comunica directamente con la BD usando sockets. Son la solución más pura desde el punto de vista JAVA. Este tipo de drivers solo los puede proporcionar el mismo vendedor de SGBD

Tipos de drivers



8. Funcionalidades de JDBC

- Creación de conexiones. Pasos a realizar
  - Cargar driver (java.sql.DriverManager)
  - Crear URL
  - Establecer la conexión (java.sql.Connection)

8. Realización de preguntas

- Un objeto “statement” es lo que envía la sentencia SQL al SGBD.
- De tres maneras dependiendo de las características de la pregunta.
  - Statement. (java.sql.Statement)
  - Prepared Statement (java.sql.PreparedStatement)
  - CallableStatement (java.sql.CallableStatement)

Lo primero que se tiene que realizar es la creación del objeto apropiado a las características del acceso a la BD.

Realización de preguntas

- Ejecución de la petición:
  - executeQuery: ejecuta un SELECT y deja el resultado en ResultSet
  - executeUpdate: ejecuta un UPDATE, INSERT o DELETE
  - Execute: ejecuta una sentencia SQL. Puede devolver un ResultSEt (si la sentencia es SELECT) o nada (si no hay más resultados o la sentencia no es un SELECT)

8. Obtención de resultados

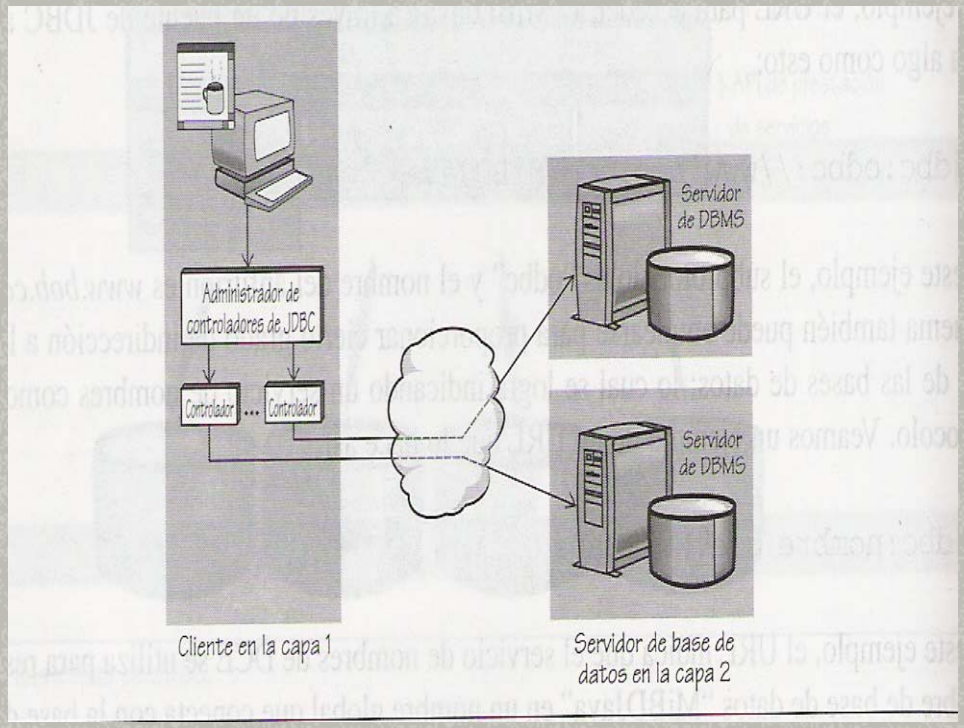
- ResultSet es el encargado de mantener las tablas resultado y de proporcionar los métodos necesarios para acceder a filas y columnas
  - Métodos:
    - *next* para mover el cursor a la siguiente tupla
    - *getXX* para obtener el valor de cada columna

    Ej.getString

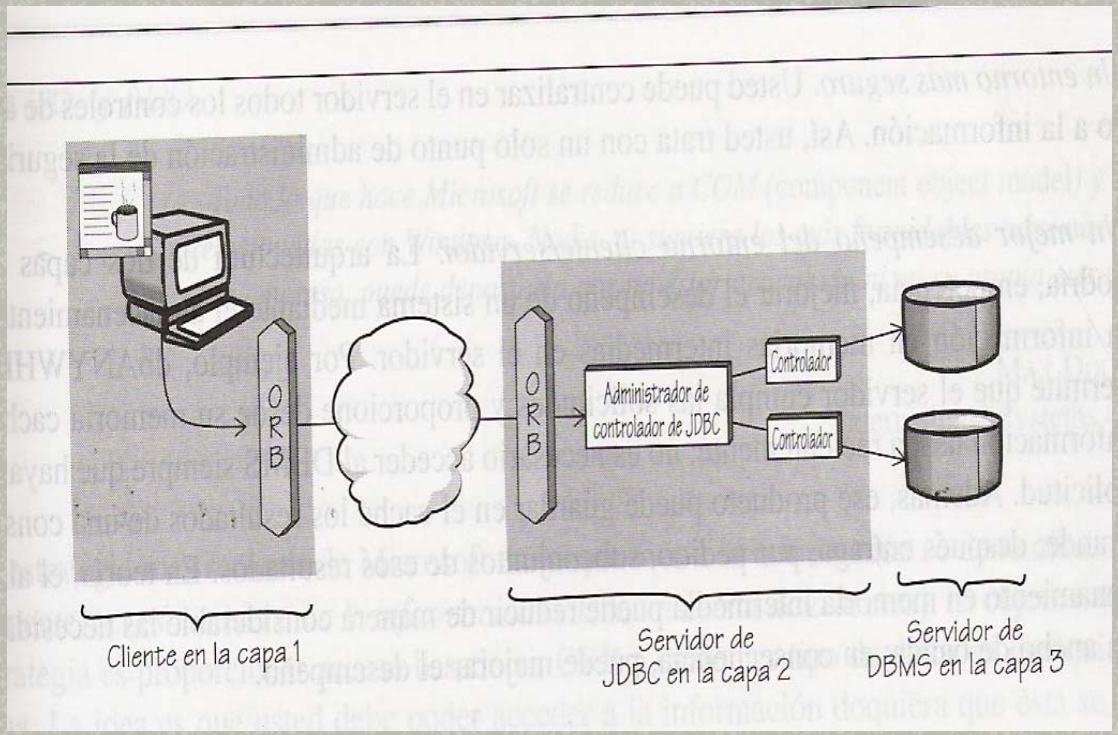
8. Cierre de la conexión y Obtención de las características de la BD.

- Para cerrar se utiliza el método close.
- Una vez establecida la conexión con la BD, se puede obtener información relativa a la misma mediante el objeto *DatabaseMeta Data.* Para crear dicho objeto se invoca al método *getMetaData* de la clase *Connection.*

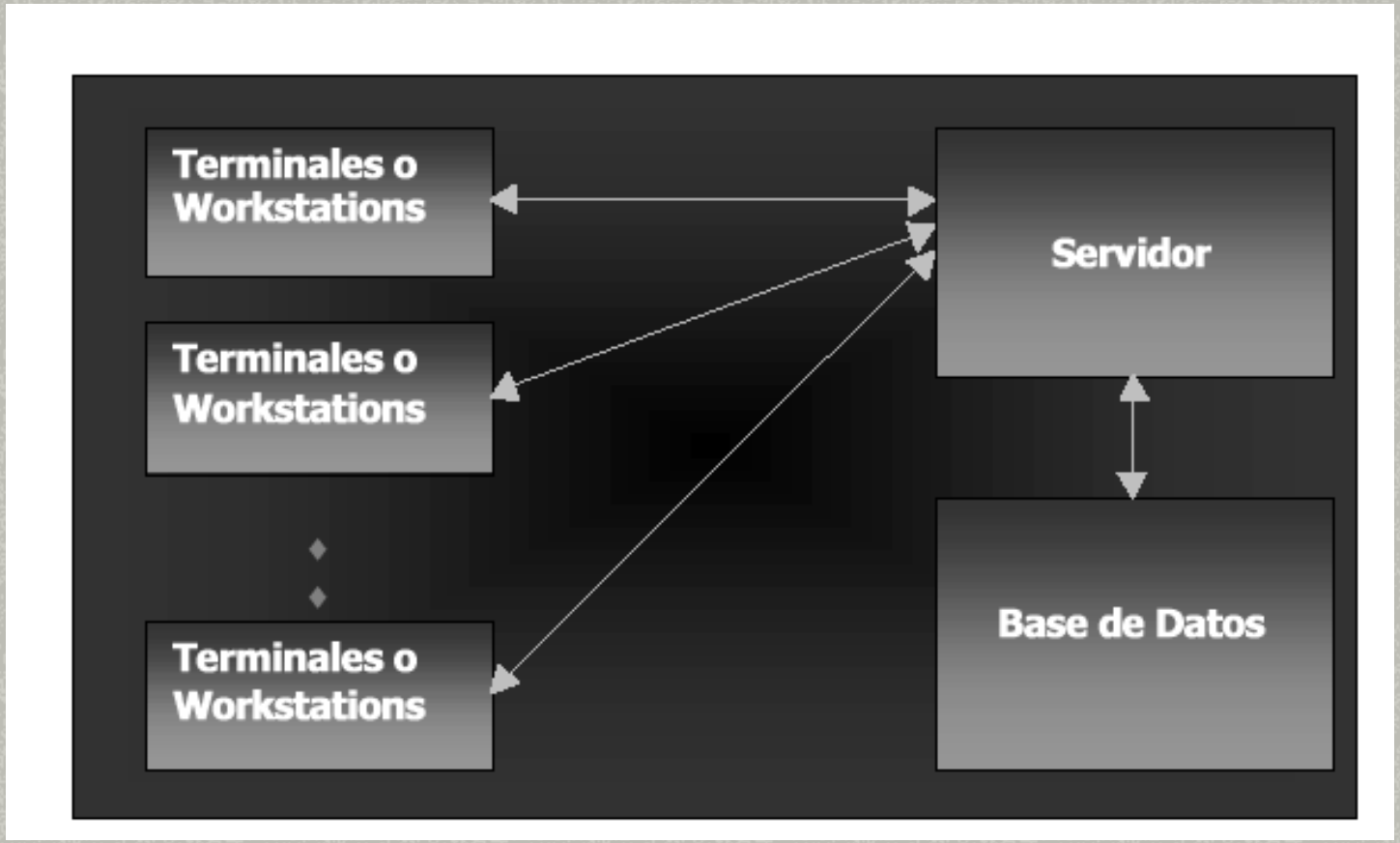
Arquitectura JDBC (2 capas)



Arquitectura JDBC (3 capas)



9. Arquitectura Cliente/Servidor de 3 niveles



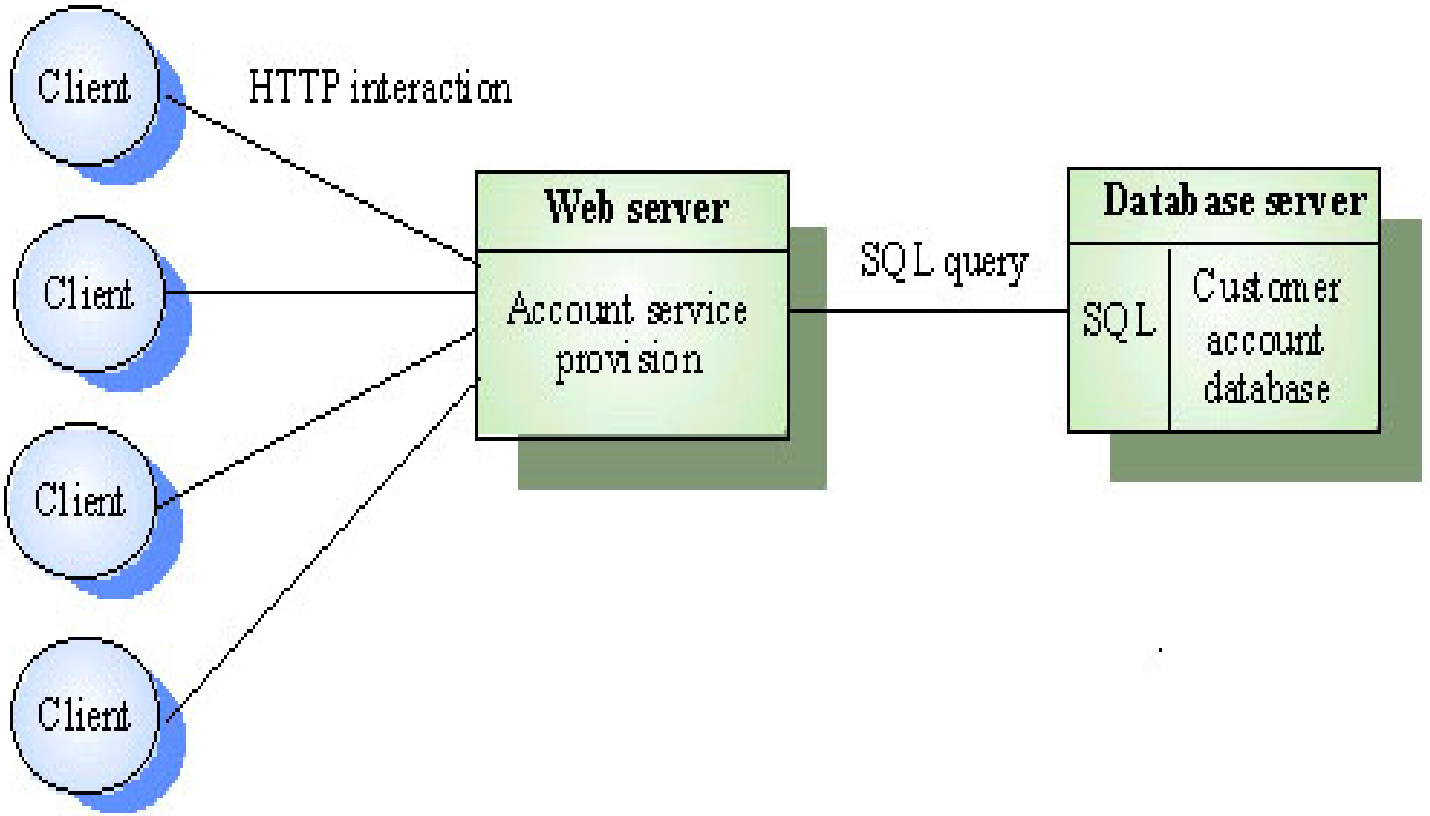
9. Arquitectura Cliente/Servidor de 3 niveles

El objetivo de este modelo es dividir las funciones de una aplicación en tres componentes:

- **Presentación:** Este componente se encarga de la interacción hombre máquina a través del monitor, teclado, ratón, o bien mediante algún otro medio como reconocedor de voz.
- **Servidores:** Compuesto por varios servidores o componentes de Software localizados en una o más plataformas que se encargará de conectar los sistemas existentes.
- **Información:** En este componente se incluye la información en sí, los sistemas y aplicaciones existentes.

Arquitectura Cliente/Servidor de 3 niveles

Arquitectura Cliente/Servidor de 3 niveles

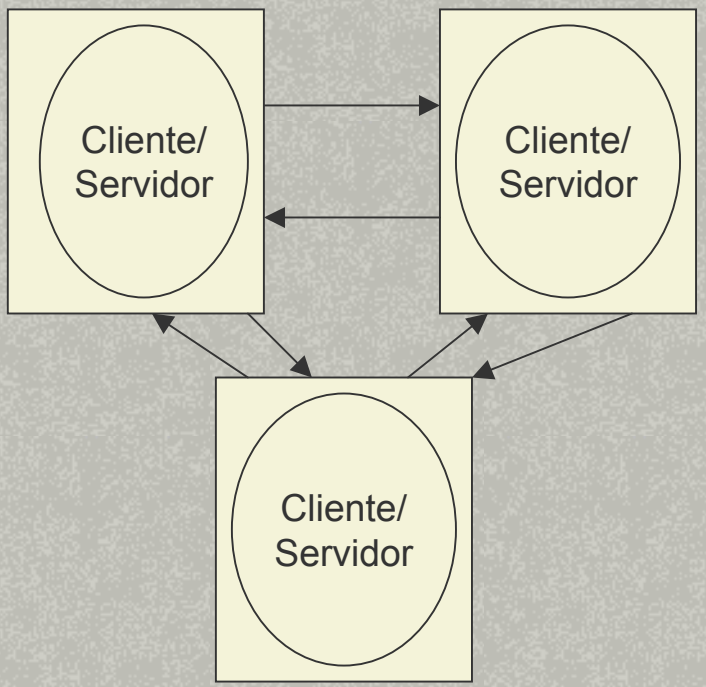


9. JSP: Java Server Pages

- Tecnología basada en Java que simplifica el desarrollo de web dinámicos.
- Ficheros de texto (extensión .jsp) que contienen etiquetas HTML y código embebido que permite al diseñador de la página web acceder a datos desde código Java que se ejecuta en el servidor.
- Cuando la página JSP es requerida por un navegador las etiquetas HTML permanecen inalterables, mientras que el código que contiene dicha página es ejecutado en el servidor, generando contenido dinámico que se combina con las etiqueta HTML antes de ser enviado al cliente.
- En http://java.sun.com/products/jsp/jsp-asp.html aparece la comparación entre JSPs y ASPs

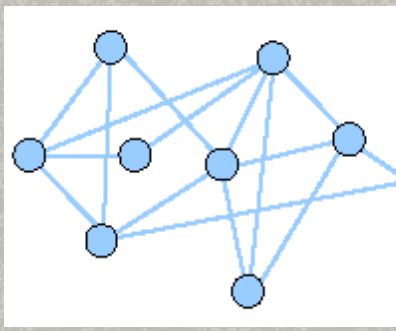
10. Peer to Peer (P2P)

- En http://lsirwww.epfl.ch/talks/ICDE2002-Tutorial.pdf se puede encontrar un tutorial sobre P2P



Todos los procesos desempeñan tareas semejantes, interactuando cooperativamente como iguales para realizar una actividad distribuida o cómputo sin distinción entre clientes y servidores

10. Peer to Peer (P2P)



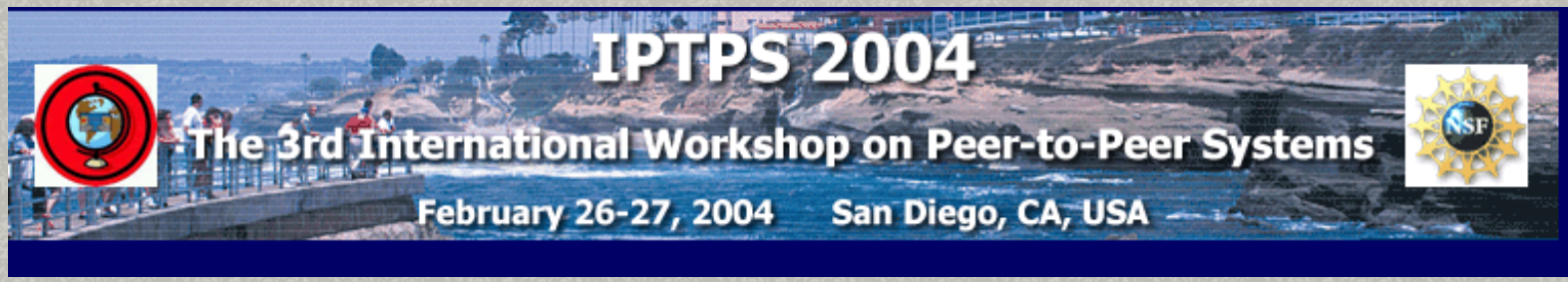
Cada nodo actúa como cliente y como servidor
Cada nodo permite el acceso a sus recursos
Propiedades:

- no existe coordinación central
- no existe una BD central
- ningún nodo tiene una visión global del sistema
- el comportamiento global surge de las iteraciones locales
- desde cualquier componente se puede acceder a todos los datos y servicios
- los componentes son autónomos
- los componentes y las conexiones son inseguras

10. Peer to Peer (P2P)

- Descentralización
  - No es necesario un único servidor
  - Los participantes pueden comunicarse directamente entre sí.
- Distribución
  - La información no está alojada en un solo sitio
- Distribución de la Carga
  - Se intenta equilibrar entre todos los participantes
- Redundancia de información
  - Se duplica información para hacerla más accesible
- Alta disponibilidad
  - La caída de un nodo no bloquea el servicio
- Optimización de uso de recursos
  - Procesamiento, almacenamiento, ancho de banda, etc…

Interés de P2P



- peer-to-peer applications and services
- peer-to-peer systems and infrastructures
- peer-to-peer algorithms
- security in peer-to-peer systems
- robustness in peer-to-peer systems
- anonymity and anti-censorship
- performance of peer-to-peer systems
- workload characterization for peer-to-peer systems
- experience with deployed peer-to-peer systems