XUNTA DE GALICIA

ESCOLA GALEGA DE ADMINISTRACIÓN PÚBLICA

26. MODELO DE CAPAS: SERVIDORES DE APLICACIONES, SERVIDORES DE DATOS, GRANJAS DE SERVIDORES. SCRIPTS DEL CLIENTE.

Fecha de actualización 26 de julio de 2011

TEMA 26. MODELO DE CAPAS: SERVIDORES DE APLICACIONES, SERVIDORES DE DATOS, GRANJAS DE SERVIDORES. SCRIPTS DEL CLIENTE.

26.1. INTRODUCCIÓN Y CONCEPTOS

En una red formada por equipos informáticos los **nodos** acostumbran a realizar tres **funciones** diferenciadas:

1. Facilitar la comunicación e interconexión de los nodos de la red.
2. Proporcionar servicios o información a otros nodos.
3. Realizar funciones de equipo de trabajo, haciendo uso de las comunicaciones, servicios e información disponible.

En este contexto los nodos o equipos que realizan las funciones de proporcionar servicio o información al resto se denominan **Servidores** y los que hacen uso de los servicios **Clientes**, formando en su conjunto lo que se da en denominar **Arquitectura Cliente-Servidor**. Se trata de una de las arquitecturas más extendidas en los entornos distribuidos, permitiendo la heterogeneidad en los clientes y un acceso transparente a la información. Los servidores permanecen a la escucha de la red en todo momento para atender las solicitudes o demandas de los clientes.

El esquema de funcionamiento **básico** seguiría el siguiente modelo:

1. El cliente solicita un servicio al servidor a través de la red.
2. El servidor a la escucha recibe la petición del servicio y la pone en la cola de demanda.
3. El servidor obtiene el resultado de la petición.
4. El servidor envía la respuesta de la petición al cliente a través de la red.
5. El cliente obtiene el resultado y lo procesa.

A partir de estos conceptos básicos podemos extraer las **características básicas** de la arquitectura Cliente-Servidor:

a) **Servicios**. Son la base de las peticiones entre los clientes y los servidores, se trata de cualquier entidad susceptible de ser demandada por uno o más clientes.
b) **Recursos compartidos.** Elementos y servicios de la red, tanto lógicos (software, datos e información), como físicos (hardware, impresoras, unidades en red, etc.).
c) **Comunicación asíncrona basada en el envío de mensajes.** Este tipo de arquitecturas emplean protocolos de comunicación asimétricos donde los clientes inician conversaciones y los servidores esperan que se establezca la comunicación escuchando la red. Toda la comunicación se realiza mediante el envío de mensajes y respuestas.
d) **Transparencia.** La localización, la organización lógica y física, así como la implementación de los servicios resulta transparente a los clientes. El uso de los mismos se limita a hacer una petición a la red y obtener la respuesta.
e) **Escalabilidad**. Horizontal en los clientes a la hora de permitir añadir nuevos nodos sin más que añadirlos a la red y vertical en los servidores, de suerte que administrando un único punto puede

mejorarse la potencia, el rendimiento, el mantenimiento y la recuperación de errores.

Fruto de la escalabilidad de esta arquitectura surgen las **granjas de servidores**, consistentes en emplear varios servidores a la vez suministrando el mismo servicio y repartiéndose las peticiones o carga del sistema. La gestión de una granja de servidores será compleja debido a la necesidad de balancear la carga para obtener el mayor rendimiento posible.

Uno de los servicios más extendidos actualmente es el web o WWW, (siglas en inglés de *World Wide Web*), se trata de un sistema de publicación e intercambio de información distribuido que relaciona unos contenidos con otros a través de vínculos. En este servicio los clientes solicitan información a modo de páginas web, tratándose de documentos en lenguajes estándar como HTML o XML que incluyen diferentes tipos de información: texto, hipervínculos, y elementos multimedia. Entre estos elementos multimedia encontramos:

a) **Texto.** Distinguiendo entre sin formato, con formato o enriquecido (tipo de letra, tamaño, color, color de fondo, etc.) e hipertexto, texto con un vínculo a otro texto o documento.
b) **Sonido**. Digitalización del habla, la música u otros sonidos.
c) **Gráficos.** Representan esquemas, planos, dibujos vectoriales, etc. son documentos que se construyen a partir de una serie de primitivas: puntos, segmentos, elipses, etc. aplicándoles a continuación todo tipo de transformaciones o funciones: giro, cambio de atributos, escalado, efectos, etc.
d) **Imágenes**. Representaciones fieles de la realidad, como fotografías. Son documentos formados exclusivamente por píxeles, punto a punto y por tanto no se estructuran o dividen en primitivas.

e) **Animación**. Representación de una secuencia de gráficos por unidad de tiempo, para ofrecer la sensación de movimiento. Asimismo ofrece posibilidades de interacción ante eventos.
f) **Video**. Representación de una secuencia de imágenes por unidad de tiempo, para ofrecer la sensación de movimiento.

Documentos complejos agrupan diversos componentes multimedia en una misma página o documento. Los sistemas de publicación actuales permiten que la multimedia **digital on line** pueda transmitirse **en flujo** (en inglés *streaming*), que se encuentra disponible tanto on line en tiempo real como bajo demanda. En este modelo no es necesario descargar o acceder a la totalidad del documento para acceder a los contenidos sino que se proporciona acceso directo a cualquier parte del flujo y reproducción desde ese punto.

Otra característica de las páginas web o documentos HTML/XML es que pueden incluir **código de script para los clientes**. Este código representa un guión o secuencia de instrucciones a manera de un programa sencillo. Este programa puede ser interpretado por el navegador del equipo cliente con unos permisos limitados en el equipo y focalizados principalmente en la página o documento web en el que se encuentran incrustados o desde el que son llamados. Existen diferentes tecnologías para estos lenguajes de script siendo las más conocidas: Javascript, Visual Basic Script, Flash, y la evolución de Javascript: AJAX, aceptados con mayor o menor fortuna por los navegadores actuales, muchas de ellas denominadas tecnologías RIA en un acercamiento de las aplicaciones web a las aplicaciones de escritorio.

26.2 MODELO DE CAPAS: SERVIDORES DE APLICACIONES,

SERVIDORES DE DATOS, GRANJAS DE SERVIDORES

La distribución de los sistemas de información fue evolucionando a lo largo del tiempo en función de las demandas y crecimiento de las redes y el aumento de la complejidad de las arquitecturas de red.

26.2.1. Arquitectura en una capa: Superordenador central.

La arquitectura más simple estaría formada por **un superordenador central** (en inglés *mainframe*) que centraliza toda la capacidad de procesamiento y almacenamiento de la red, también denominada monolítica. En este modelo el acceso a la información se hace directamente a través de la computadora principal o bien a través de clientes ligeros que se limitan a hacer las funciones de terminales. En esta arquitectura centraliza todo el coste de administración y mantenimiento se dedica al servidor central. Los terminales carecen de programas propios y tienen recursos de memoria o disco mínimos, pudiendo incluso carecer de disco. Cualquier instalación o avance en el servidor repercute al punto en la red de suerte que cualquier programa instalado estará disponible para todos los clientes. Por contra, si tenemos en cuenta la sostenibilidad del sistema en caso de caída o errores en el servidor central toda la red se ve afectada, al igual que si un cliente sobrecarga el sistema todos los demás se verán afectados en cuanto a rendimiento. A su vez, los mainframes se organizan según arquitecturas paralelas tipo **SNA** (en inglés *Systems Network Architecture*) con un diseño de red con comunicación P2P a través de APPN (en inglés *Advanced Peer-to-Peer Networking*).

26.2.2. Arquitectura en dos capas: Modelo Cliente-Servidor.

En este modelo el sistema se estructura en dos capas, una capa a nivel de usuario que almacena y procesa parte de la información y otra capa remota

a nivel de servicios que almacena y da funcionalidad a la totalidad de los clientes de la red. De este modo se consigue descargar de parte de la carga de la red a los servidores centrales y mantiene la capa de servicios transparente a los usuarios con la posibilidad de escalar el sistema mejorando o aumentando el número de servidores sin que estos lleguen a notar el cambio en algo más que el rendimiento. Por contra, un modelo más distribuido en lo relativo a los clientes obliga a un mayor mantenimiento de los mismos por parte de los administradores. Otro de los puntos a tener en cuenta es la consistencia de los datos entre cliente y servidor, de suerte que hace falta coordinar cada servicio por separado.

En esta línea el uso de protocolos de comunicación soporta el uso efectivo por parte de los clientes de los servicios de la red permitiendo la heterogeneidad de los clientes siempre y cuando los implementen.

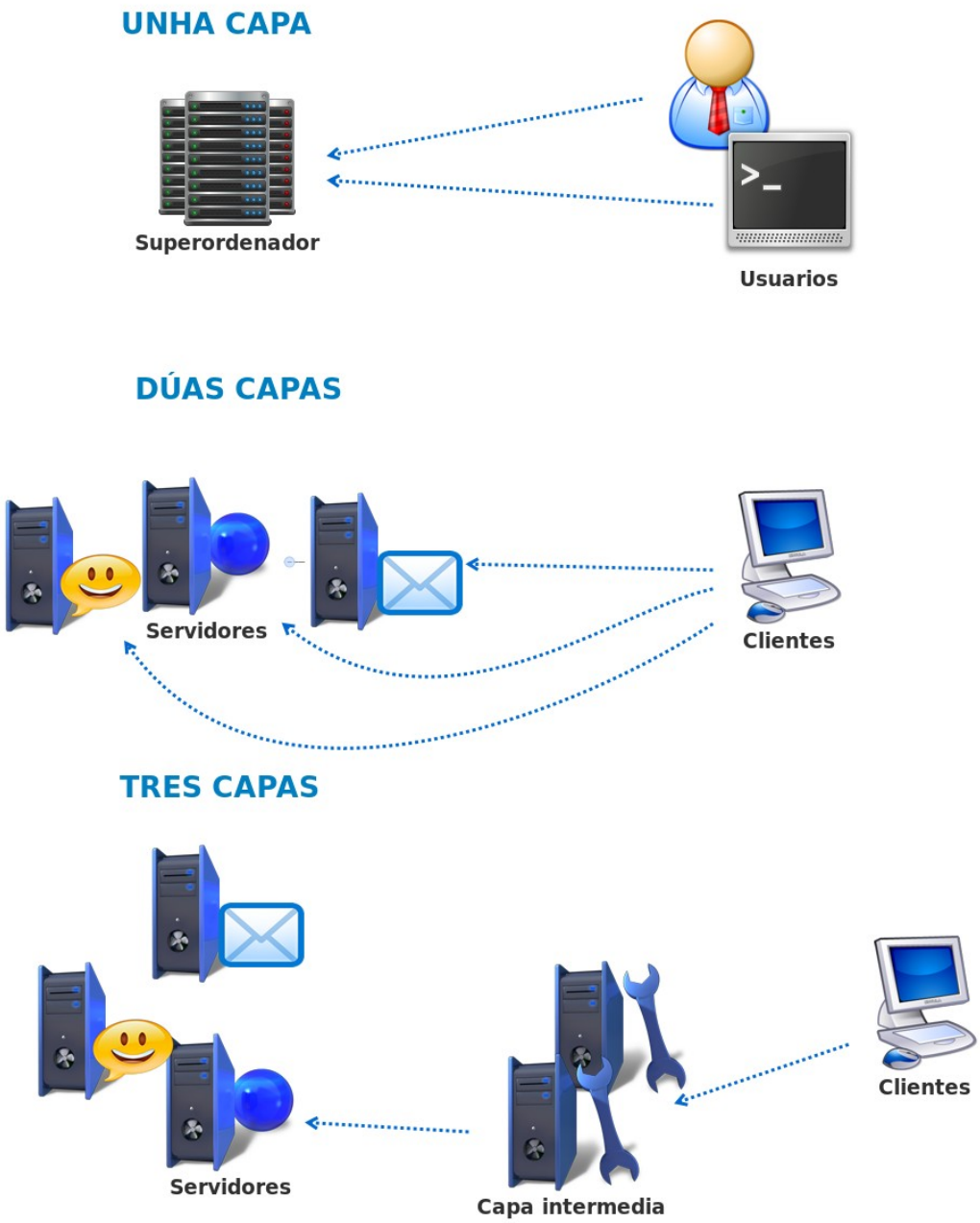


Figura 1: Arquitecturas en capas

26.2.3. Arquitectura en tres capas: Granjas de servidores.

Debido a los inconvenientes de los sistemas de una única capa, que obligan a mantener un servidor central de tamaño demasiado grande para un mantenimiento y rendimiento eficientes, y de dos capas, que obligan a

mantener cada servidor independiente del resto para un único servicio, se optó por el establecimiento de una capa más entre las de cliente y servidor. En esta capa se agrupan varios servidores en una DMZ soportando el mismo servicio dando lugar a una redundancia que tiene como ventajas una mayor tolerancia a fallos y un avance de rendimiento. A efectos de la red las granjas de servidores proporcionan un único servicio lógico o virtual integrado por cualquier número de servidores físicos. Cada servidor de la granja debe ser una réplica exacta del servidor lógico en cuanto a datos y software instalado. Para escalar el sistema se añade a la granja un nuevo servidor réplica del virtual y aumenta la disponibilidad de recursos. El ejemplo más habitual de granja de servidores es un servicio web, donde si por ejemplo un servidor atiende a mil usuarios y tenemos previstos picos de diez mil usuarios simultáneos pondremos una granja de diez servidores, para atender el servicio y otros dos más en previsión de caída de alguno o picos puntuales aún más altos. La escalabilidad de las granjas en función de los servicios ofertados define tipologías básicas como *Datacenters*, servidores de aplicaciones, de importación, de *front-end* existiendo elementos específicos para control de carga, Teredo o de dominio, entre otros.

26.2.3.1 Componentes intermedios: *Middleware*

Para dar el efecto de transparencia a los clientes, ese sistema requiere de una serie de componentes intermedios, es decir que se encuentran “por el medio” (en inglés *middleware*) de las capas principales. Estos componentes se encargan de recibir y repartir las peticiones de los clientes entre los servidores de la granja, cuidar el balanceo de carga, el mantenimiento de la sesión, etc. El ***middleware*** se describe como un conductor o intermediario entre sistemas, dirigiendo las peticiones de datos y servicios a otros nodos de la red. Entre sus principales características destacarían:

a) Simplificar el desarrollo de aplicaciones al capsular comunicaciones

entre sistemas.

b) Facilitar la interconexión de los sistemas de información con independencia de la red física.

c) Mejorar la escalabilidad del sistema, aumentando la capacidad sin pérdida de funcionalidad.

d) Mejorar la tolerancia a fallos del sistema, fiabilidad.

e) Aumentar la complejidad de administración y soporte.

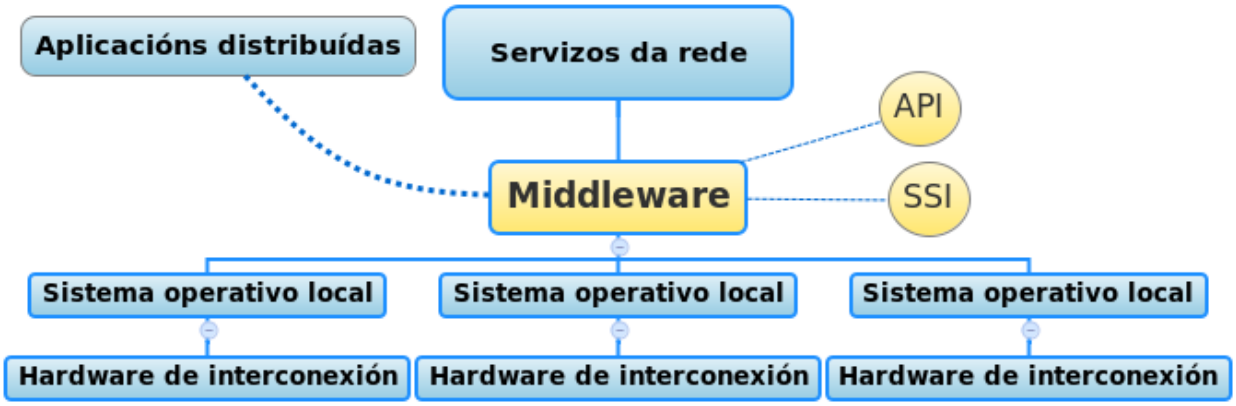


Figura 2: El middleware en un sistema distribuido.

TRADUCCIÓN TEXTO FIGURA 2: Aplicaciones distribuidas. Servicios de la red.

En un sistema distribuido el *middleware* es el software de conectividad que permite disponer de un conjunto de servicios sobre plataformas distribuidas heterogéneas. Actúa como una capa de abstracción de las funciones del sistema distribuido haciendo transparente en la red los sistemas operativos y el hardware de interconexión de las redes de comunicaciones. Proporciona una Interfaz de Programación de Aplicación (**API**) para la comunicación y acceso a aplicaciones y servicios distribuidos. Por otra parte proporciona una interfaz única de acceso al sistema denominada **SSI** (del inglés *Single System Image*), la cual da al cliente la sensación de acceder a un único servidor, el virtual.

Para garantizar la heterogeneidad en la comunicación de los sistemas, el *middleware* se estructura en tres **capas o niveles de comunicación** separados:

1) **Protocolo de transporte.** Protocolos de comunicaciones comunes a la capa de transporte de la red, como TCP o UDP. Establecen niveles de seguridad, control de sesiones, etc.
2) **Sistema Operativo en Red o NOS** (en inglés *Network Operating System*). Extensión del sistema operativo de los clientes que captura las peticiones y las dirige hacia el servidor adecuado para devolver a continuación la respuesta del mismo al cliente.
3) **Protocolo de servicio.** Protocolo específico del servicio o aplicación en el sistema Cliente-Servidor.

Los middleware acostumbran a clasificarse según el tipo de comunicación que realizan en el Sistema Operativo en Red y a los parámetros que comunican (infraestructura, acceso a datos, aplicaciones, etc.), los **tipos de middleware** más habituales serían:

1. **Llamadas a procedimientos remotos** (en inglés *Remote Procedure Call* o RPC). Los Clientes invocan directamente procedimientos o funciones de procesos que se ejecutan en servidores remotos, permitiendo distribuir la lógica de la aplicación remota a través de la red. Las llamadas pueden realizarse de manera asíncrona o síncrona. Mantiene al mínimo la información de la sesión y en caso de ruptura de la misma el cliente reinicia la comunicación de cero.
2. **Publicación/suscripción.** Este middleware realiza una monitorización del sistema detectando los servicios y procesos activos. Los componentes registran su interés en determinados eventos, en cuanto estos eventos son detectados por el monitor

envía esa información a los suscriptores. La interacción es asíncrona recayendo por completo en el servicio de notificación/monitorización.

3. **Middleware orientado a mensajes** (en inglés *Message Oriented Middleware* o MOM). La comunicación se basa en el envío de mensajes asíncronos por parte de los nodos (cliente, servidor, servicio o aplicación). Los mensajes se recogen en colas priorizadas en el nodo destino y se almacenan hasta que pueden responderse. El funcionamiento del sistema es análogo a cómo funciona un servicio de correo electrónico.
4. **Middleware basado en objetos** (en inglés *Object Request Broker* u ORB). Incorpora a RPC los paradigmas de orientación a objetos. Define una arquitectura cliente servidor donde los servicios devuelven objetos, siendo estos la unidad de comunicación. Los nodos piden los objetos por el nombre siendo estos entregados por un servicio de resolución de nombres. Ejemplos de implementaciones de este *middleware* serían: CORBA, RMI, COM, .NET Remoting, etc.
5. **Middleware de acceso a datos** (en inglés *Oriented Data Access Middleware*). Proporcionan la API transparente de acceso a datos agrupando las operaciones de manejo de la conexión con las bases de datos. Ejemplos de este tipo de API serían JDBC y ODBC. Por norma general realizan conexiones síncronas y operaciones transaccionales.
6. **Arquitecturas orientadas a servicios** (en inglés *Service Oriented Architecture* o SOA). Las funcionalidades o procedimientos se publican desde cualquier servidor a modo de servicios. Los servidores publican el servicio y permanecen a la escucha hasta que llega una petición, la procesan y devuelven una respuesta al cliente del servicio. Ejemplos de este *middleware* son los Servicios Web y los Servicios CORBA.

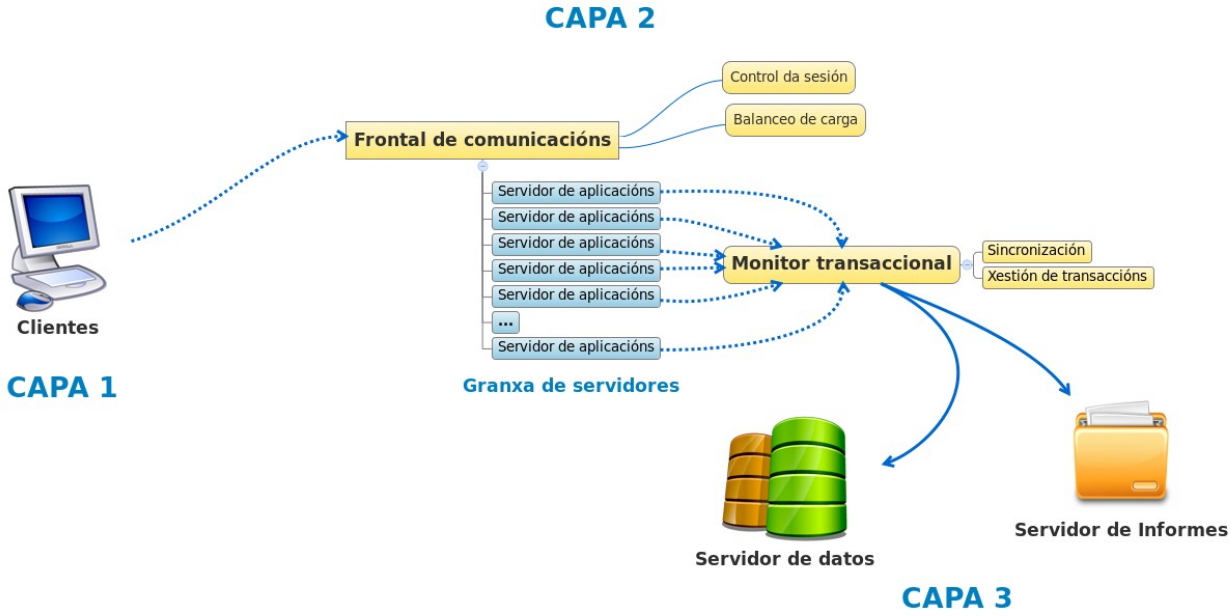


Figura 3: Granja de servidores.

TRADUCCIÓN TEXTOS FIGURA 3: Servidor de aplicaciones. Gestión de transacciones. Granja de servidores.

Las granjas de servidores se completan con dos componentes fundamentales, funciones que de manera general realiza el *middleware*:

1. **El frontal de comunicaciones.** El frontal de comunicaciones (en inglés *front-end*) es el punto de acceso único a la granja de servidores, simulando un único servidor lógico. Aunque cada servidor de la granja pueda tener su propia dirección IP lo normal es que el frontal tenga uno propio y sea ésta la forma en que los clientes accedan a él, o bien directamente o bien a través de un nombre de dominio (DNS).
   - ✓ **Balanceo de carga.** Consiste en dividir la carga de trabajo de los clientes entre los servidores de la granja. Puede implementarse vía hardware, software o una combinación de ambas. Para hacerlo vía hardware el frontal de comunicaciones debe disponer de un

equipo específico, aunque hay enrutadores que permiten esta funcionalidad.

- ✓ **Control de la sesión**. Si debido al balanceo de la carga diferentes peticiones de un mismo cliente van hacia servidores diferentes hace falta coordinar a los servidores en el seguimiento de una sesión o que puedan compartirla.
- ✓ **Priorización**. En caso de tener peticiones simultáneas el frontal debe ser capaz de atender primero a los clientes críticos o de mayor prioridad así como de asignar el procesamiento de sus tareas a aquellos servidores dotados de más recursos.

2. **Los monitores transaccionales.** Los monitores transaccionales son los encargados de mantener en las redes la consistencia de los datos y los procesos que se realizan simultáneamente en los servidores de la granja. Tiene que garantizar que una modificación de datos fruto de una petición se realiza como una transacción, es decir, o se realiza completamente o no se realiza en absoluto. Cada transacción tiene que tener lugar independientemente de que tenga lugar otra simultánea, deben procesarse de manera aislada. Las principales funciones serán:
   - ✓ **La gestión de las transacciones.** Se encarga de controlar la atomicidad y secuencialidad de las transacciones, para garantizar la consistencia de datos y operaciones. La gestión de transacciones debe garantizar el correcto funcionamiento del sistema cuando la carga de trabajo o el número de usuarios son muy elevados. Debe contemplar la posibilidad de errores en las aplicaciones y caídas de elementos del sistema durante las transacciones, permitiendo operación de vuelta atrás. Vista detenidamente la gestión de transacciones constará de:
     1. **Gestor de transacciones** (en inglés *Transaction Manager*).

Controla el inicio de transacción, registra los recursos que precisa y gestiona las operaciones de confirmación de la transacción (en inglés *commit*) y de vuelta atrás y recuperación del estado inicial de la transacción (en inglés *rollback*).

2. **Gestor de registro** (en inglés *Log Manager*). Guardar los estados de los recursos que están en uso por parte de las transacciones, elaborando un historial de versiones de los mismos. Esta información es compartida por los distintos gestores de transacciones siendo lo que permite garantizar la consistencia de los recursos empleados.
3. **Gestor de bloqueos** (en inglés *Lock Manager*). Gestiona el acceso simultáneo por parte de varios procesos a los recursos, permitiendo bloquearlos para evitar que dos o más accesos a la vez den lugar a inconsistencias. Asimismo lleva a cabo la detección del momento en que se libera un recurso y envía una notificación al gestor de transacciones.

- ✓ **Sincronización**. La sincronización de las comunicaciones resulta compleja en este modelo, ya que un cliente puede acceder a un servicio empleando diferentes servidores de la capa intermedia, incluso simultáneamente. Según el comportamiento del servicio podemos encontrar soluciones síncronas, donde se espera siempre a la respuesta del servidor, simple pero con riesgo de bloqueo. Frente a las asíncronas donde se envía la petición y ya llegará la respuesta, con lo cual no hay bloqueos, pero la respuesta podría no llegar nunca sin más opción que detectarlo a través de tiempos de espera agotados.

26.2.6. Arquitecturas en n-Capas.

Las arquitecturas en tres capas pueden extenderse a n-capas cuando en la

capa intermedia se incorporan otros elementos de interconexión como distribuidores o sistemas de cortafuego. También puede dividirse la capa de servidores por servicios o diferentes capas de acceso a datos o presentación de información. La separación en capas es una organización lógica del sistema con el que se puede establecer cualquier número de capas según las necesidades del mismo. Cualquier especialización de servidores que se quiera hacer en la red y provoque un nuevo agrupamiento podemos identificarla con una nueva capa.

26.2.7. Arquitecturas para Red entre iguales (P2P).

En los modelos distribuidos de igual a igual o P2P (en inglés *Peer-to-Peer*), todos los equipos (excepto los elementos de interconexión) tienen el mismo papel doble de cliente/servidor en la red. Hacen uso de servicios y los proporcionan. En esta arquitectura por tanto no se pueden agrupar los nodos y pierde sentido hablar de capas. Estas redes no resultan óptimas para todo tipo de servicios, en muchos casos, por ejemplo a la hora de funcionar como servidor web, requerirían un coste muy alto para control de la consistencia del sitio web, mantenimiento y configuración del servidor, etc. Por el contrario, en situaciones donde los nodos caen a menudo, por ejemplo un servidor atacado continuamente, la redundancia de nodos garantiza que el sistema siga funcionando. Otros servicios como el intercambio de archivos, o el procesamiento compartido presentan más ventajas a la hora de emplear una arquitectura de este tipo como solución de implantación. Los modelos más habituales son centralizados, puros, descentralizados o híbridos, según el peso de cada nodo individual en la red o de la existencia de servidores con responsabilidad de control y gestión en el modelo. La adaptación de este modelo por parte de los ISP da lugar a P2P híbridas de servicio denominadas P4P (en inglés *Proactive network Provider Participation for P2P*). Otros modelo similares serían los

P2M, que actúan en arquitecturas híbridas empleando el correo electrónico como soporte del envío de datos.

La gestión de este tipo de redes se realiza fundamentalmente vía software. En este caso el software deberá realizar las mismas funciones ya vistas para la arquitectura de tres o más capas: frontal de comunicaciones y gestión de transacciones. Por debajo acostumbran implementar servidores propios como Kademlia, eDonkey, Gnutella, FastTrack, BitTorrent u OpenNap entre otros.

26.2.8. SERVIDORES.

26.2.8.1. Servidor web.

Se trata de servidores que proveen el servicio WWW a través del protocolo HTTP. En esencia se trata de una aplicación ejecutándose en un servidor a la espera de peticiones HTTP por parte de un cliente respondiendo con los documentos solicitados generalmente páginas web y los objetos que enlazan: imágenes, archivos de script , animaciones, etc. En funciones más avanzadas estos servidores añaden seguridad a través de conexiones encriptadas con protocolos tipo HTTP Seguro o HTTPS. Por regla general, los servidores de aplicaciones se integran en arquitecturas de mínimo tres **capas**:

1) **Primera capa.** Capa de interacción con los usuarios, principalmente a través de navegadores web.
2) **Capa intermedia.** Capa de los servidores web, que pueden estar distribuidos en un modelo de granja de servidores. Cada servidor incorporaría los módulos necesarios para seguridad, lenguajes de servidor interpretados, correo, mensajería, acceso a datos y otras funcionalidades.
3) **Tercera capa.** Capa de servidores de acceso a datos, como

servidores de archivos, base de datos o informes.

Entre los **servidores web de uso más extendido** actualmente se encontrarían:

- ✓ **Apache**. Uno de los más utilizados, por ser un servidor libre que ofrece prestaciones a nivel de otras soluciones propietarias además de una gran facilidad de uso y configuración.
- ✓ **Internet Information Server (IIS).** Servidor propietario con soporte para aplicaciones .NET o ASP entre otras.
- ✓ **Otros:** Java Web Server, AOLServer, Cherokee, Tomcat, lightHttpd, etc.

Los servidores web pueden disponer de módulos para la ejecución de programas de servidor interpretados, como son los de las tecnologías Python, PHP, ASP, JSP, Tcl, etc.

26.2.8.2. Servidor de aplicaciones.

Los servidores de aplicaciones son servidores web con capacidad de procesamiento ampliada, pudiendo ejecutar aplicaciones y componentes de lógica de negocio y recursos relacionados como el acceso a datos. Debido a esto permiten realizar el procesamiento de aplicaciones de cliente en el propio servidor. Proporcionan soporte como middleware o software de conectividad y para diferentes tecnologías de servidor. Por regla general, los servidores de aplicaciones se integran en arquitecturas de mínimo tres **capas**:

1) **Primera capa.** Capa de interacción con los usuarios, principalmente a través de navegadores web.
2) **Capa intermedia.** Capa de los servidores de aplicaciones, que pueden estar distribuidos en un modelo de granja de servidores. Un subconjunto de los servidores de aplicaciones darán servicio a los

usuarios/clientes mientras otro grupo se encargará de soportar la operativa común del dominio, como librerías o aplicaciones y servicios web de los que hagan uso las aplicaciones para usuarios/clientes.

3) **Tercera capa.** Capa de servidores de acceso a datos, como servidores de archivos, base de datos o informes.

El servidor de aplicaciones acostumbra a tener integrado un servidor web, para gestionar de manera independiente el servicio WWW a través del protocolo HTTP.

Además de este servicio presenta un amplio conjunto de herramientas:

- ✓ Servidor web integrado.
- ✓ Contenedor de programas de servidor (en inglés *servlets*).
- ✓ Contenedores de objetos de lógica de negocio (como por ejemplo EJBs).
- ✓ Sistemas de mensajería.
- ✓ Software de conectividad con bases de datos.
- ✓ Balanceo de carga.
- ✓ Gestión de límites y colas de conexiones (en inglés *Pool*) para bases de datos y objetos.
- ✓ Etc.

En esencia un servidor de aplicaciones realiza las mismas funciones que un servidor web, pero cuando la demanda de uso es grande y estamos ante un sistema complejo la solución pasa por emplear un servidor de aplicaciones que ofrezca las siguientes **ventajas**:

- ✓ **Centralización**. Centraliza en los servidores la administración y configuración de la lógica de negocio de las aplicaciones, de manera que aspectos como el mantenimiento de los accesos a base de datos pueden realizarse de manera centralizada. Asimismo cambios

derivados de actualizaciones, migraciones o recuperaciones ante errores tienen lugar desde un único punto.

- ✓ **Seguridad.** Al existir un único punto de acceso a datos puede reforzarse la defensa y los sistemas de control de errores en ese punto, mejorando su gestión y protección.
- ✓ **Rendimiento.** Como punto intermedio permite gestionar las peticiones de los clientes a la Base de datos.
- ✓ **Escalabilidad.** Un mismo servidor de aplicaciones puede dar servicio a varios clientes, y por tanto aumentando el número de servidores se mejora el rendimiento del sistema.

Entre los **servidores de uso más extendido** actualmente se encontrarían:

- ✓ **Jboss, Glassfish.** Servidores de aplicaciones libres bajo licencia GPL.
- ✓ **BEA Weblogic, IBM Websphere, Oracle Application Server.** Alternativas propietarias integradas en paquetes de aplicaciones con funcionalidades de gestión y monitorización extendidas.
- ✓ **Tomcat, Internet Information Server (IIS), Jetty**. Proporcionan funciones parciales de servidores de aplicaciones, con lo cual en ocasiones se definen más bien como contenedores de programas de servidor.

26.2.8.3. Servidor de acceso a datos.

Los servidores de acceso a datos ocuparían la última capa de los sistemas de información encargándose del acceso directo a los datos, existiendo diferentes tipos según el sistema de información empleado para su almacenamiento o publicación:

- ✓ **Servidores de archivos.** En este tipo de servidores la información se almacena directamente en archivos, por tanto la función de estos equipos será la de permitir el acceso remoto a los mismos desde los

clientes u otros servidores. Los protocolos más habituales ofrecen servicio solo desde redes locales pero en sistemas avanzados pueden proporcionar servicios como FTP o WebDAV para conexión remota a través de Internet. Actualmente el término empleado para referirse a estos servidores es NAS (en inglés *Network-Attached Storage*), pero ésta tan sólo sería la tecnología más habitual frente a otras como DAS (en inglés *Direct Attached Storage*), basada en SCSI o SAN (en inglés *Storage Arena Network*) basada en fibra óptica. No requieren un software muy específico como otros tipos de servidores sino más bien soporte para diferentes protocolos y tecnologías. Por norma general acostumbran a estar dispuestos en RAID (en inglés *Redundant Arrays of Independent Disks*), equipos de almacenamiento redundante.

- ✓ **Servidores de bases de datos.** Albergan uno o más sistemas de gestión de bases de datos (en inglés *database management system*, o DBMS), software de gestión que se encarga de la comunicación entre las aplicaciones y las bases de datos. Permiten realizar operaciones de definición, manipulación y seguridad de los datos a través de una API de comunicación con las aplicaciones y un lenguaje estructurado de consulta como el SQL. Permiten accesos simultáneos a los datos, seguridad y gestión de transacciones.

Estos sistemas suelen presentar además programas o consolas de administración avanzadas para realizar las tareas generales de gestión de la base de datos.

- ✓ **Servidores de informes.** Pueden considerarse una capa intermedia entre los servidores de datos y los de aplicación, donde se establecen servidores o granjas de servidores que sirven los datos en documentos predefinidos multiformato: hojas de cálculo, PDF, XML, HTML, etc. El software de este tipo de servidores acostumbra a incorporar software de gestión para el servidor, y software de auto-edición de informes, para definir modelo de informes compuestos de

cabeceras, imágenes, fórmulas, subinformes, etc. que se generarán dinámicamente los informes a partir de consultas sobre los datos.

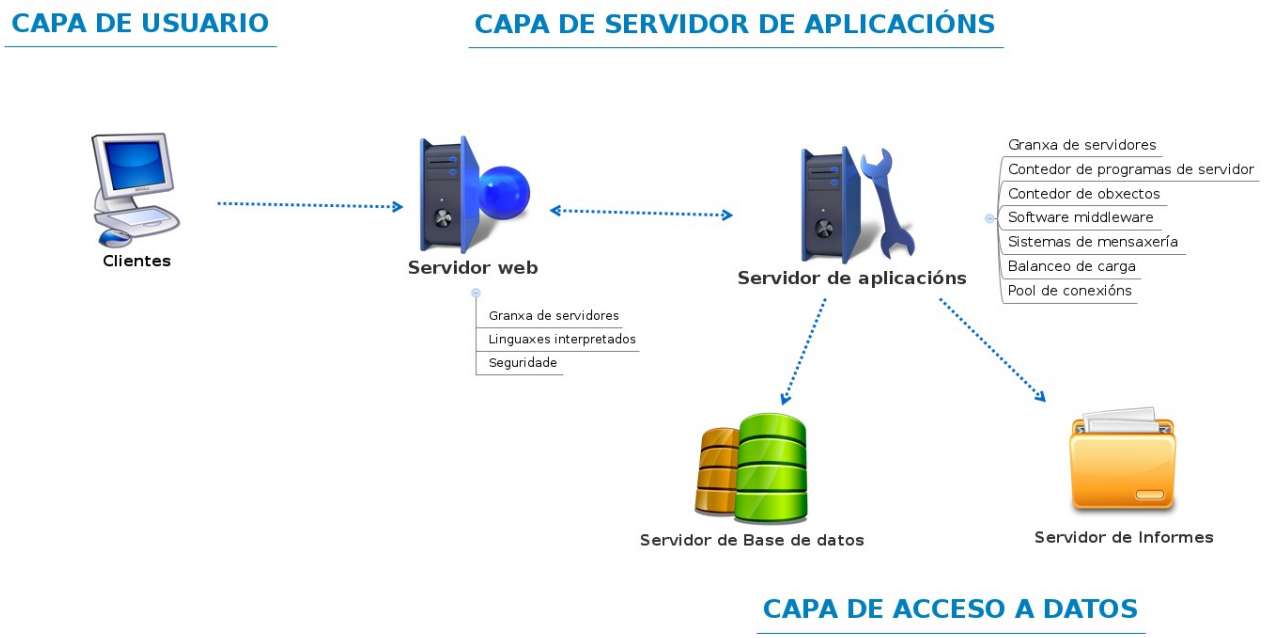


Figura 4: Arquitectura en 3 capas con servidor web, de aplicaciones y base de datos.

TRADUCCIÓN FIGURA 4: Capa de servidor de aplicaciones. Granja de servidores. Lenguajes interpretados. Seguridad. Contenedor de programas de servidor. Contenedor de objetos. Sistemas de mensajería. Pool de conexiones.

Entre los **servidores de uso más extendido** actualmente se encontrarían:

- ✓ **Servidores de archivos.** No requieren gestores especializados pero sí soporte software a los protocolos: CIFS, NFS, SMB, FTP, WebDAV, etc. Así como utilidades tipo Samba o FreeNAS.
- ✓ **Servidores de bases de datos.**
  - ✓ De licencia libre: PostgreSQL, MariaDB, Firebird, SQLite, Apache derby,...
  - ✓ Dual, dependiendo de su uso: MySQL.

- ✓ Software propietario: SQLServer, Oracle, Access, Paradox, Informix, DBase, etc.
- ✓ **Servidores de informes.** Jasper Reports, Jreports, Crystal Reports, Oracle Reports etc.

26.3 SCRIPTS DEL CLIENTE.

Los *scripts* del cliente son programas interpretados diseñados para ejecutarse en los navegadores con el objetivo de dotar a las páginas de mayor interactividad con el usuario y dinamismo en una aproximación a las aplicaciones de escritorio. El **funcionamiento básico** de un *script* consiste en interpretar una serie de comandos a través de los cuales puede modificar y manipular objetos y reaccionar ante eventos de la interfaz como respuestas a periféricos (ratón, teclado, etc.), o cambios en los elementos del documento (botones, elementos de formularios, etc.). Sus usos básicos son validaciones, manipulación de formularios, procesamiento de funciones y carga asíncrona de datos.

Los *scripts* de cliente proporcionan las siguientes **ventajas**:

- ✓ Modificar el contenido de la página sin recargarla del servidor en función de las interacciones con el usuario.
- ✓ Modificar parámetros de configuración del navegador y otros elementos de la página web.
- ✓ Mejorar la interacción entre el usuario y el documento, en general la usabilidad.

Por el contrario, presentan una serie de inconvenientes o desventajas:

- ✓ Problemas de accesibilidad, pues se complica la posibilidad de

presentar alternativas a usuarios que no soporten la tecnología de script.

- ✓ Problemas de seguridad, pues toda la lógica de interacción aparece sin protección descargada en el equipo del usuario, con lo cual disponen del código fuente del programa de script.

Las **tecnologías de *script*** más empleadas son Visual Basic Script, Javascript, con su evolución AJAX y PerlScript. El principal problema a la hora de seleccionar una tecnología cuando se diseña una página web es el soporte que recibirá por parte de los navegadores, pues hay que recordar que los lenguajes de *script* serán en última instancia interpretados en el navegador.

a) **Javascript**. Basado en el lenguaje Java, es una de los lenguajes de script de uso más extendido. Hay que señalar que tiene aplicación con otras tecnologías además de la web, como en documentos PDF o aplicaciones de escritorio. Para permitir la interacción con los elementos de un documento web este lenguaje dispone de una API que implementa el DOM (en inglés *Document Object Model*) o Modelo de Objetos para la Representación del Documento, estandarizado por el W3C (en inglés *World Wide Web Consortium*). En un intento por estandarizar este lenguaje surge el ECMAScript una especificación del lenguaje aceptado cómo estándar ISO.

b) **Visual Basic Script.** Similar al Javascript en lo relativo a funcionamiento y estructura pero basado en Visual Basic. Tiene menor soporte dentro de los diferentes navegadores, excepto en el Internet Explorer.

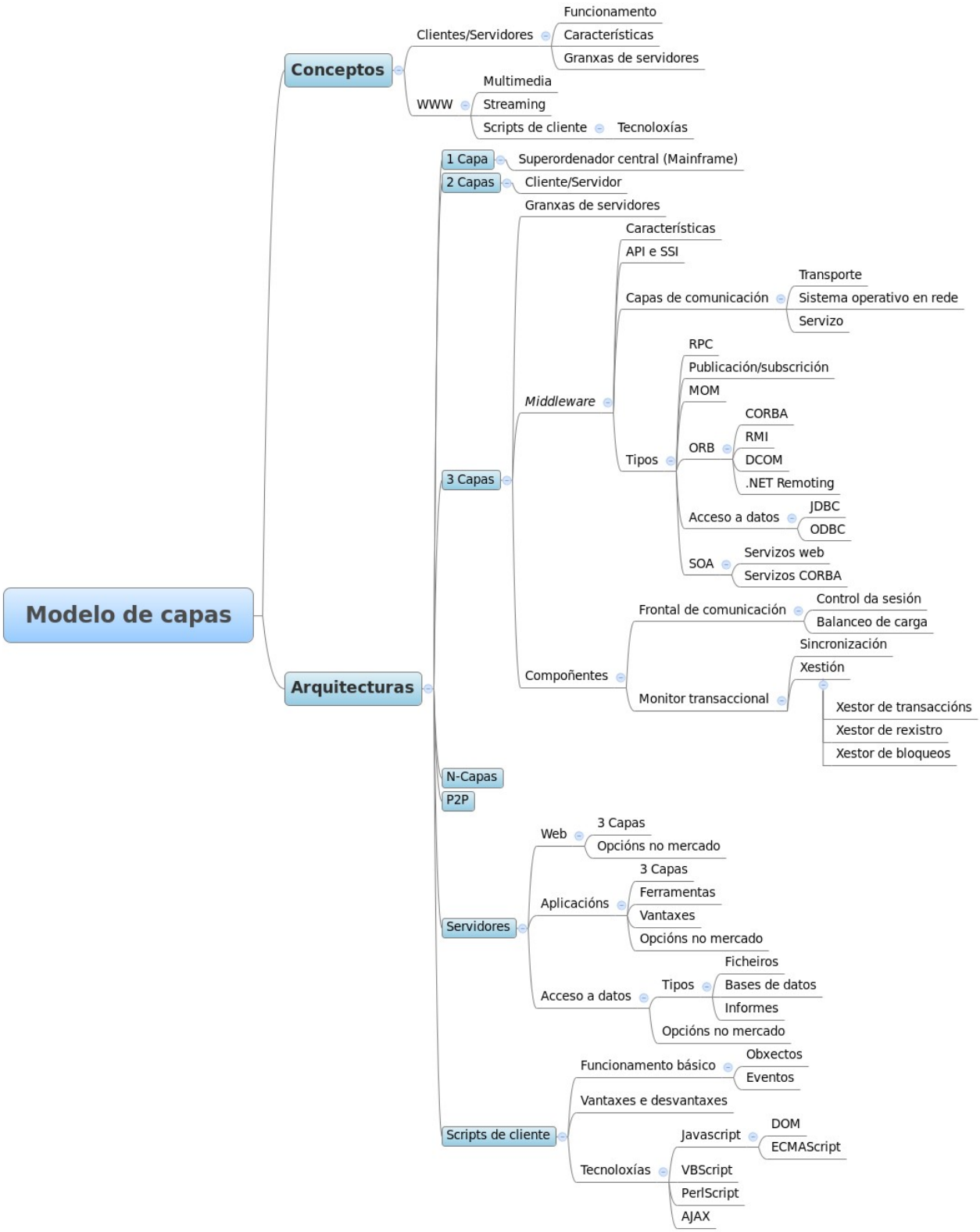
c) **PerlScript**. Basado en lenguaje C su uso no está tan extendido como las tecnologías anteriores aunque tiene menos limitaciones. Debido a esto fue derivando hacia el lenguaje de servidor.

d) **AJAX**. Acrónimo de Javascript Asíncrono y XML (en inglés

Asynchronous Javascript And XML). Esta es una tecnología de *script* de cliente asíncrona, de suerte que puede realizar cargas de datos sin que afecten a la recarga de la página. AJAX es un conjunto de tecnologías que hace uso de:

1) XHTML y hojas de estilo en cascada (CSS) para la estructura y diseño de los contenidos.
2) El lenguaje Javascript como lenguaje de programación para funciones y definición del programa cliente.
3) El objeto *XMLHttpRequest* para intercambio de información asíncrona con el servidor. Por tanto, hace falta que el navegador soporte este objeto, siendo empleado en ocasiones el objeto *Iframe*.
4) XML y DOM como estándares asociados para intercambio de datos y manipulación del documento.

26.4. ESQUEMA


Modelo de capas
Conceptos
Clientes/Servidores
Funcionamento
Características
Granxas de servidores
WWW
Multimedia
Streaming
Scripts de cliente
Tecnoloxías
Arquitecturas
1 Capa
Superordenador central (Mainframe)
2 Capas
Cliente/Servidor
3 Capas
Granxas de servidores
Middleware
Características
API e SSI
Capas de comunicación
Transporte
Sistema operativo en rede
Servizo
Tipos
RPC
Publicación/subscrición
MOM
ORB
CORBA
RMI
DCOM
.NET Remoting
Acceso a datos
JDBC
ODBC
SOA
Servizos web
Servizos CORBA
Compoñentes
Frontal de comunicación
Control da sesión
Balanceo de carga
Monitor transaccional
Sincronización
Xestión
Xestor de transaccións
Xestor de rexistro
Xestor de bloqueos
N-Capas
P2P
Servidores
Web
3 Capas
Opcións no mercado
Aplicacións
3 Capas
Ferramentas
Vantaxes
Opcións no mercado
Acceso a datos
Tipos
Ficheiros
Bases de datos
Informes
Opcións no mercado
Scripts de cliente
Funcionamento básico
Obxectos
Eventos
Vantaxes e desvantaxes
Tecnoloxías
Javascript
DOM
ECMAScript
VBScript
PerlScript
AJAX

TRADUCCIÓN ESQUEMA: Granjas de servidores. Tecnologías. Sistema operativo en red. Publicación /Suscripción. Servicios web. Servicios CORBA. Componentes. Gestión. Gestor de transacciones. Gestor de registro. Gestor de bloqueos. Opciones en el mercado. Aplicaciones. Herramientas. Ventajas. Ficheros- Funcionamiento básico. Objetos. Ventajas y desventajas. Tecnologías

26.5. REFERENCIAS

José Antonio Mañas.
Mundo IP. Introducción a los secretos de Internet y las redes de datos. (2004).

Andrew S. Tanenbaum.
Redes de computadoras. (2003).

Sergio Luján Mora.
Programación de aplicaciones web: historia, principios básicos y clientes web. (2003).

Autor: Juan Marcos Filgueira Gomis
Asesor Técnico Consellería de Educación e O. U.
Colegiado del CPEIG