TEMARIO OPOSICIÓNS INFORMÁTICA

GRUPO A2 – XESTIÓN E SISTEMAS DE INFORMACIÓN

TEMA 26. MODELO DE CAPAS: SERVIDORES DE APLICACIÓNS, SERVIDORES DE DATOS, GRANXAS DE SERVIDORES. SCRIPTS DO CLIENTE.

Autor: Juan Marcos Filgueira Gomis

TEMA 26. MODELO DE CAPAS: SERVIDORES DE APLICACIÓNS, SERVIDORES DE DATOS, GRANXAS DE SERVIDORES. SCRIPTS DO CLIENTE.

26.1. INTRODUCIÓN E CONCEPTOS
26.2 MODELO DE CAPAS: SERVIDORES DE APLICACIÓNS, SERVIDORES DE DATOS, GRANXAS DE SERVIDORES.
26.3 SCRIPTS DO CLIENTE.
26.4. ESQUEMA
26.5. REFERENCIAS

26.1. INTRODUCIÓN E CONCEPTOS

Nunha rede formada por equipos informáticos os **nodos** acostuman realizar tres **funcións** diferenciadas:

1. Facilitar a comunicación e interconexión dos nodos da rede.
2. Proporcionar servizos ou información a outros nodos.
3. Realizar funcións de equipo de traballo, facendo uso das comunicacións, servizos e información dispoñible.

Neste contexto os nodos ou equipos que realizan as funcións de proporcionar servizo ou información ao resto denomínanse **Servidores** e os que fan uso dos servizos **Clientes**. Formando no seu conxunto o que se da en denominar **Arquitectura Cliente-Servidor**. Trátase dunha das arquitecturas máis estendidas nos contornos distribuídos, permitindo a heteroxeneidade nos clientes e un acceso transparente á información. Os servidores permanecen á escoita da rede en todo momento para atender ás solicitudes ou demandas dos clientes.

O esquema de **funcionamento básico** seguiría o seguinte modelo:

1. O cliente solicita un servizo ao servidor a través da rede.
2. O servidor á escoita recibe a petición do servizo e a pon na cola de demanda.
3. O servidor obtén o resultado da petición.
4. O servidor envía a resposta da petición ao cliente a través da rede.
5. O cliente obtén o resultado e o procesa.

A partir destes conceptos básicos podemos extraer as **características básicas** da arquitectura Cliente-Servidor:

a) **Servizos**. Son a base das peticións entre os clientes e o servidores, trátase de calquera entidade susceptible de ser demandada por un ou máis clientes.
b) **Recursos compartidos.** Elementos e servizos da rede, tanto lóxicos (software, datos e información), como físicos (hardware, impresoras, unidades en rede, etc...).
c) **Comunicación asíncrona baseada no envío de mensaxes.** Este tipo de arquitecturas empregan protocolos de comunicación asimétricos onde os clientes inician conversas e os servidores esperan que se estableza a comunicación escoitando a rede. Toda a comunicación se realiza mediante o envío de mensaxes e respostas.
d) **Transparencia.** A localización, a organización lóxica e física, así como a implementación dos servizos resulta transparente aos clientes. O uso dos mesmos limítase a facer unha petición á rede e obter a resposta.
e) **Escalabilidade**. Horizontal nos clientes á hora de permitir engadir novos nodos sen máis que engadilos á rede e vertical nos servidores, de xeito que administrando un único punto pode mellorarse a potencia, o rendemento, o mantemento e a recuperación de erros.

Froito da escalabilidade desta arquitectura xorden as **granxas de servidores**, consistentes en empregar varios servidores á vez subministrando o mesmo servizo e repartíndose as peticións ou carga do sistema. A xestión dunha granxa de servidores será complexa debido á necesidade de balancear a carga para obter o maior rendemento posible.

Un dos servizos máis estendidos actualmente é o web ou **WWW**, (siglas en inglés de *World Wide Web*), trátase dun sistema de publicación e intercambio de información distribuído que relaciona uns contidos con outros a través de ligazóns. Neste servizo os clientes solicitan información a modo de páxinas web, tratándose de documentos en linguaxes estándar como HTML ou XML que inclúen diferentes tipos de información: texto, hiperligazóns, e elementos multimedia. Entre estes elementos multimedia atopamos:

a) **Texto.** Distinguindo entre sen formato, con formato ou enriquecido (tipo de letra, tamaño, cor, cor de fondo, etc...) e hipertexto texto cun vínculo ou ligazón a outro texto ou documento.
b) **Son**. Dixitalización da fala, a música ou outros sons.
c) **Gráficos.** Representan esquemas, planos, debuxos vectoriais, etc... son documentos que se constrúen a partires dunha serie de primitivas: puntos, segmentos, elipses, etc... aplicándolles a continuación todo tipo de transformacións ou funcións: rotación, cambio de atributos, escalado, efectos, etc...
d) **Imaxes**. Representacións fieis da realidade, como fotografías. Son documentos formados exclusivamente por píxeles, punto a punto e por tanto non se estruturan ou dividen en primitivas.
e) **Animación**. Representación dunha secuencia de gráficos por unidade de tempo, para ofrecer a sensación de movemento. Así mesmo ofrece posibilidades de interacción ante eventos.
f) **Vídeo**. Representación dunha secuencia de imaxes por unidade de tempo, para ofrecer a sensación de movemento.

Documentos complexos agrupan diversos compoñentes multimedia nunha mesma páxina ou documento. Os sistemas de publicación actuais permiten que a **multimedia dixital en liña** poda transmitirse **en fluxo** (en inglés *streaming*), que se atopa dispoñible tanto en liña en tempo real coma baixo demanda. Neste modelo non é necesario descargar ou acceder á totalidade do documento para acceder aos contidos senón que se proporciona acceso directo a calquera parte do fluxo e reprodución dende ese punto.

Outra característica das páxinas web ou documentos HTML/XML é que poden incluír **código de *script* para os clientes**. Este código representa un guión ou secuencia de instrucións a xeito dun programa sinxelo. Este programa pode ser interpretado polo navegador do equipo cliente cuns permisos limitados no equipo e focalizados principalmente na páxina ou documento web no que se atopan incrustados ou dende o que son chamados. Existen diferentes tecnoloxías para estas linguaxes de *script* sendo as máis coñecidas: Javascript, Visual Basic Script, Flash, e a evolución de Javascript: AJAX, aceptados con maior ou menor fortuna polos navegadores actuais, moitas delas denominadas tecnoloxías RIA nun achegamento das aplicacións web ás aplicacións de escritorio.

26.2 MODELO DE CAPAS: SERVIDORES DE APLICACIÓNS, SERVIDORES DE DATOS, GRANXAS DE SERVIDORES

A distribución dos sistemas de información foi evolucionando ao longo do tempo en función das demandas e crecemento das redes e o aumento da complexidade das arquitecturas de rede.

26.2.1. Arquitectura nunha capa: Superordenador central.

A arquitectura máis simple estaría formada por un **superordenador central** (en inglés *mainframe*) que centraliza toda a capacidade de procesamento e almacenamento da rede, tamén denominada monolítica. Neste modelo o acceso á información faise directamente a través da computadora principal ou ben a través de clientes lixeiros que se limitan a facer as funcións de terminais. Nesta arquitectura centraliza todo o custe de administración e mantemento adícase ao servidor central. Os terminais carecen de programas propios, e teñen recursos de memoria ou disco mínimos, podendo mesmo carecer de disco. Calquera instalación ou mellora no servidor repercute ao momento na rede de xeito que calquera programa instalado estará dispoñible para todos os clientes. Por contra, se temos en conta a sostibilidade do sistema en caso de caída ou erros no servidor central toda a rede vese afectada, do mesmo xeito que se un cliente sobrecarga o sistema todos os demais veranse afectados en canto a rendemento. Á súa vez os mainframes organizánse segundo arquitecturas paralelas tipo **SNA** (en inglés *Systems Network Architecture*) cun deseño de rede con comunicación P2P a través de **APPN** (en inglés *Advanced Peer-to-Peer Networking*).

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

Neste modelo o sistema se estrutura en dúas capas, unha capa a nivel de usuario que almacena e procesa parte da información e outra capa remota a nivel de servizos que almacena e da funcionalidade á totalidade de clientes da rede. Deste xeito conséguese descargar de parte da carga da rede aos servidores centrais e mantén a capa de servizos transparente aos usuarios coa posibilidade de escalar o sistema mellorando ou aumentando o número de servidores sen que estes cheguen a notar o cambio en algo máis que o rendemento. Por contra, un modelo máis distribuído no tocante aos clientes obriga a un maior mantemento dos mesmos por parte dos administradores. Outro dos puntos a ter en conta é a consistencia dos datos entre cliente e servidor, de xeito que cómpre coordinar cada servizo por separado.

Nesta liña o uso de protocolos de comunicación soporta o uso efectivo por parte dos clientes dos servizos da rede permitindo a heteroxeneidade dos clientes sempre e cando os implementen.

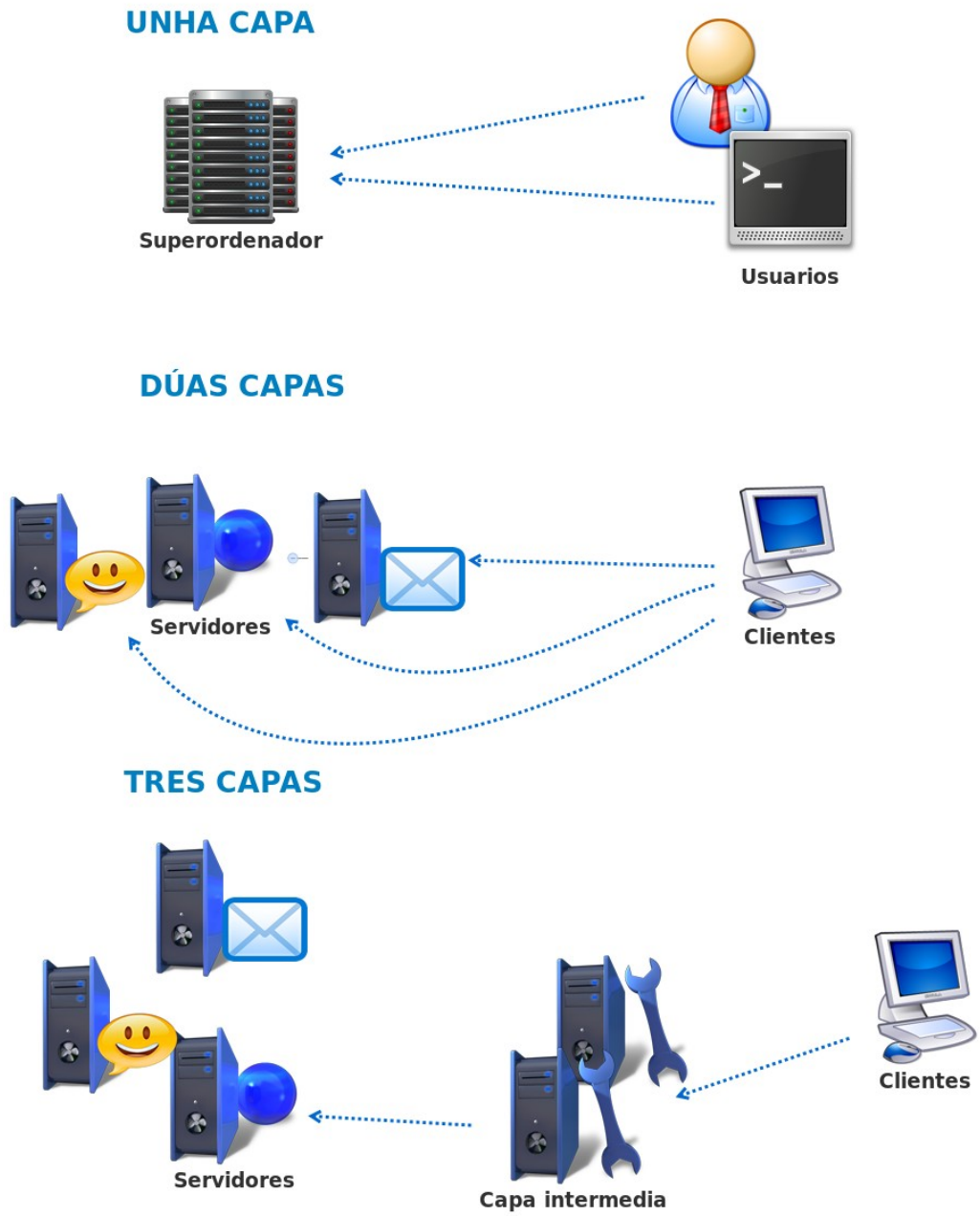


Figura 1: Arquitecturas en capas

26.2.3. Arquitectura en tres capas: Granxas de servidores.

Por mor dos inconvenientes dos sistemas dunha única capa, que obrigan a manter un servidor central de tamaño demasiado grande para un mantemento e rendemento eficientes, e de dúas capas, que obrigan a manter cada servidor independente do resto para un único servizo, optouse polo establecemento dunha capa máis entre as de cliente e servidor.

Nesta capa agrúpanse varios servidores nunha DMZ soportando o mesmo servizo dando lugar a unha redundancia que ten como vantaxes unha maior tolerancia a fallos e unha mellora de rendemento. A efectos da rede a granxa de servidores proporcionan un único servizo lóxico ou virtual integrado por calquera número de servidores físicos. Cada servidor da granxa debe ser unha réplica exacta do servidor lóxico en canto a datos e software instalado. Para escalar o sistema engádese á granxa un novo servidor réplica do virtual e aumenta a dispoñibilidade de recursos. O exemplo máis habitual de granxa de servidores é un servizo web, onde se por exemplo un servidor atende a mil usuarios e temos previstos picos de dez mil usuarios simultáneos poremos unha granxa de dez servidores, para atender o servizo e outros dous máis en previsión de caída dalgún ou picos puntuais aínda máis altos. A escalabilidade das granxas en función dos servizos ofertados define tipoloxías básicas como *Datacenters*, servidores de aplicacións, de importación, de *front-end* existindo elementos específicos para control de carga, Teredo ou de dominio, entre outros.

26.2.3.1 Compoñentes intermedios: *Middleware*

Para dar o efecto de transparencia aos clientes, ese sistema require dunhas serie de compoñentes intermedios, é dicir que se atopan "polo medio" (en inglés *middleware*) das capas principais. Estes compoñentes se encargan de recibir e repartir as peticións dos clientes entre os servidores da granxa, coidar o balanceo de carga, o mantemento da sesión, etc... O ***middleware*** descríbese coma un condutor ou intermediario entre sistemas, dirixindo as peticións de datos e servizos a outros nodos da rede. Entre as súas principais características destacarían:

a) Simplificar o desenvolvemento de aplicacións ao capsular comunicacións entre sistemas.
b) Facilitar a interconexión dos sistemas de información con independencia da rede física.
c) Mellorar a escalabilidade do sistema, aumentando a capacidade sen perda de funcionalidade.
d) Mellorar a tolerancia a fallos do sistema, fiabilidade.
e) Aumentar a complexidade de administración e soporte.

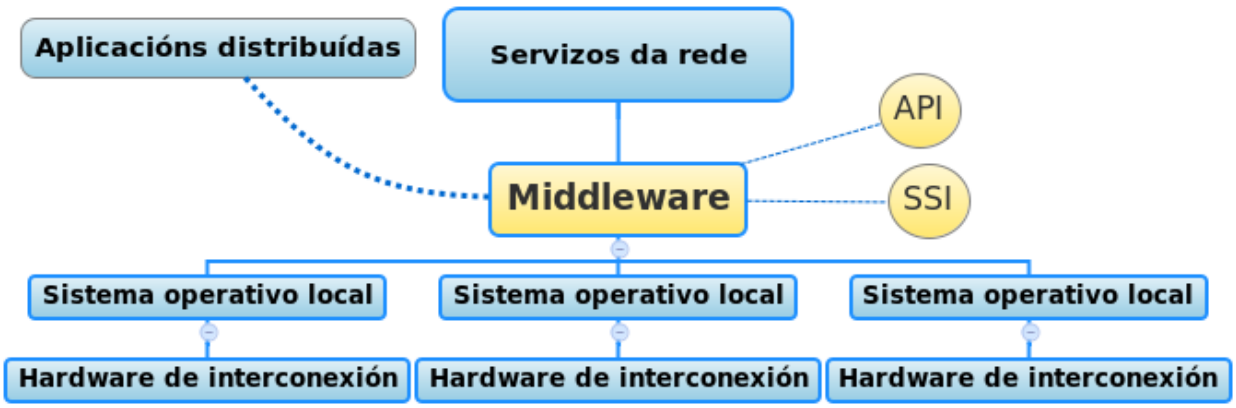


Figura 2: O middleware nun sistema distribuído.

Nun sistema distribuído o *middleware* é o software de conectividade que permite dispoñer dun conxunto de servizos sobre plataformas distribuídas heteroxéneas. Actúa coma unha capa de abstracción das funcións do sistema distribuído facendo transparente na rede os sistemas operativos e o hardware de interconexión das redes de comunicacións. Proporciona unha Interface de Programación de Aplicación (**API**) para a comunicación e acceso a aplicacións e servizos distribuídos. Por outra banda proporciona unha interface única de acceso ao sistema denominada **SSI** (do inglés *Single System Image*), a cal da ao cliente a sensación de acceder a un único servidor, o virtual.

Para garantir a heteroxeneidade na comunicación dos sistemas o *middleware* estruturase en tres **capas ou niveis de comunicación** separados:

1) **Protocolo de transporte.** Protocolos de comunicacións comúns á capa de transporte da rede, como TCP ou UDP. Establecen niveis de seguridade, control de sesións, etc...
2) **Sistema Operativo en Rede ou NOS** (en inglés *Network Operating System*). Extensión do sistema operativo dos clientes que captura as peticións e as dirixe cara o servidor axeitado para devolver a continuación a resposta do mesmo ao cliente.
3) **Protocolo de servizo.** Protocolo específico do servizo ou aplicación no sistema Cliente-Servidor.

Os middleware acostuman a clasificarse segundo o tipo de comunicación que realizan no Sistema Operativo en Rede e aos parámetros que comunican (infraestrutura, acceso a datos, aplicacións, etc...), os **tipos de middleware** máis habituais serían:

1. **Chamadas a procedementos remotos** (en inglés *Remote Procedure Call* ou RPC). Os Clientes invocan directamente procedementos ou funcións de procesos que se executan en servidores remotos, permitindo distribuír a lóxica da aplicación remota a través da rede. As chamadas poden realizarse de maneira asíncrona ou síncrona. Mantén ao mínimo a información da sesión e en caso de ruptura da mesma o cliente reinicia a comunicación de cero.
2. **Publicación/subscrición.** Este middleware realiza unha monitorización do sistema detectando os servizos e procesos activos. Os compoñentes rexistran o seu interese en determinados eventos en cando estes eventos son detectados polo monitor envía esa información aos subscritores. A interacción é asíncrona recaendo por completo no servizo de notificación/monitorización.
3. **Middleware orientado a mensaxes** (en inglés *Message Oriented Middleware* ou MOM). A comunicación basease no envío de mensaxes asíncronos por parte dos nodos (cliente, servidor, servizo ou aplicación). As mensaxes recóllense en colas priorizadas no nodo destino e almacénanse ata que poden responderse. O funcionamento do sistema é análogo a como funciona un servizo de correo electrónico.
4. **Middleware baseado en obxectos** (en inglés *Object Request Broker* ou ORB). Incorpora a RPC os paradigmas de orientación a obxectos. Define unha arquitectura cliente servidor onde os servizos devolven obxectos, sendo estes a unidade de comunicación. Os nodos piden os obxectos polo nome sendo estes entregados por un servizo de resolución de nomes. Exemplos de implementacións deste *middleware* serían: CORBA, RMI, COM, .NET Remoting, etc...
5. **Middleware de acceso a datos** (en inglés *Oriented Data Access Middleware*). Proporcionan a API transparente de acceso a datos agrupando as operación de manexo da conexión coas bases de datos. Exemplos deste tipo de API serían JDBC e ODBC. Por norma xeral realizan conexións síncronas e operacións transaccionais.
6. **Arquitecturas orientadas a servizos** (en inglés *Service Oriented Architecture* ou SOA). As funcionalidades ou procedementos se publican dende calquera servidor a modo de servizos. Os servidores publican o servizo e permanecen á escoita ata que chega unha petición, a procesan e devolven unha resposta ao cliente do servizo. Exemplos deste middleware son os Servizos Web e os Servizos CORBA.

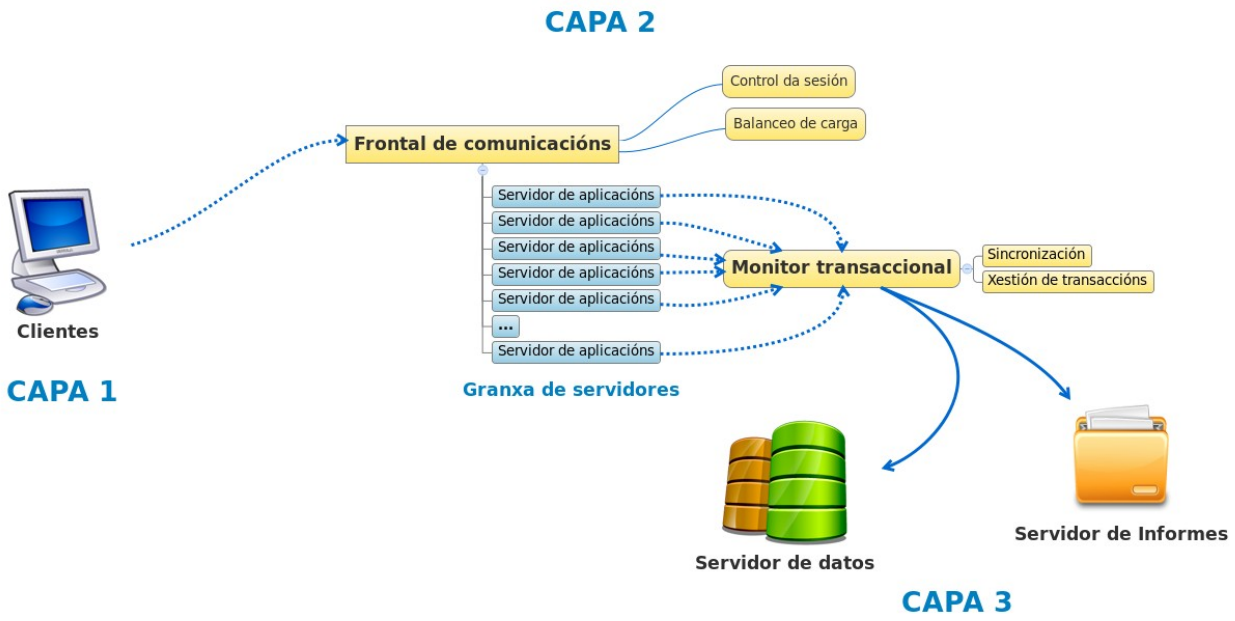


Figura 3: Granxa de servidores.

As granxas de servidores complétanse con dous compoñentes fundamentais, funcións que de xeito xeral realiza o *middleware*:

1. **O frontal de comunicacións.** O frontal de comunicacións (en inglés *front-end*) é o punto de acceso único á granxa de servidores, simulando un único servidor lóxico. Aínda que cada servidor da granxa poida ter o seu propio enderezo IP o normal é que o frontal teña un propio e sexa esta a forma de que os clientes accedan a el, ou ben directamente ou ben a través dun nome de dominio (DNS).
   - ✔ **Balanceo de carga.** Consistir en dividir a carga de traballo dos clientes entre os servidores da granxa. Pode implementarse vía hardware, software ou unha combinación de ambas. Para facelo vía hardware o frontal de comunicacións debe dispoñer dun equipo específico, aínda que hai enrutadores que permiten esta funcionalidade.
   - ✔ **Control da sesión**. Se por mor do balanceo da carga diferentes peticións dun mesmo cliente van cara servidores diferentes cómpre coordinar aos servidores no seguimento dunha sesión ou que poidan compartila.
   - ✔ **Priorización**. En caso de ter peticións simultáneas o frontal debe ser capaz de atender primeiro aos clientes críticos ou de maior prioridade así como de asignar o procesamento das súas tarefas a aqueles servidores dotados de máis recursos.

2. **Os monitores transaccionais.** Os monitores transaccionais son os encargados de manter a de redes consistencia dos datos e os procesos que se procesan simultaneamente nos servidores da granxa. Ten que garantir que unha modificación de datos froito dunha petición se realiza como unha transacción, é dicir, ou se realiza completamente ou non se realiza en absoluto. Cada transacción ten que ter lugar independente de que teña lugar outra simultánea, deben procesarse de xeito illado. As principais funcións serán:
   - ✔ **A xestión das transaccións.** Encárgase de controlar a atomicidade e secuencialidade das transaccións, para garantir a consistencia de datos e operacións. A xestión de transaccións debe garantir o correcto funcionamento do sistema cando a carga de traballo ou o número de usuarios son moi elevados. Debe contemplar a posibilidade de erros nas aplicacións e caídas de elementos do sistema durante as transaccións, permitindo operación de volta atrás. Vista polo miúdo a xestión de transaccións constará de:
     1. **Xestor de transaccións** (en inglés *Transaction Manager*). Controla o inicio de transacción, rexistra os recursos que precisa e xestiona as operacións de confirmación da transacción (en inglés *commit*) e de volta atrás e recuperación do estado inicial da transacción (en inglés *rollback*).
     2. **Xestor de rexistro** (en inglés *Log Manager*). Gardar os estado dos recursos que están uso por parte das transaccións, elaborando un historial de versións dos mesmos. Esta información é compartida polos distintos xestores de transaccións sendo o que permite garantir a consistencia dos recursos empregados.
     3. **Xestor de bloqueos** (en inglés *Lock Manager*). Xestiona o acceso simultáneo por parte de varios procesos aos recursos, permitindo bloquealos para evitar que dous ou máis accesos á vez dean lugar a inconsistencias. Así mesmo leva a cabo a detección de cando se libera un recurso e envía unha notificación ao xestor de transaccións.
   - ✔ **Sincronización**. A sincronización das comunicacións resulta complexa neste modelo, xa que un cliente pode acceder a un servizo empregando diferentes servidores da capa intermedia, mesmo simultaneamente. Segundo o comportamento do servizo podemos atopar solucións síncronas, onde se espera sempre á resposta do servidor, simple pero con risco de bloqueo. Fronte as asíncronas onde se envía a petición e xa chegará a resposta, co cal non hai bloqueos, pero a resposta podería non chegar nunca sen máis opción que detectalo a través de tempos de espera esgotados.

26.2.6. Arquitecturas en n-Capas.

As arquitecturas en tres capas poden estenderse a n-capas cando na capa intermedia incorpóranse outros elementos de interconexión como distribuidores ou sistemas de devasa. Tamén pode dividirse a capa de servidores por servizos ou diferentes capas de acceso a datos ou presentación de información. A separación en capas é unha organización lóxica do sistema co cal pode establecerse calquera número de capas segundo as necesidades do mesmo. Calquera especialización de servidores que se queira facer na rede e provoque un novo agrupamento podemos identificala cunha nova capa.

26.2.7. Arquitecturas para Rede entre iguais (P2P).

Nos modelos distribuídos de igual a igual ou **P2P** (en inglés *Peer-to-Peer*), todos os equipos (agás os elementos de interconexión) teñen o mesmo rol dobre de cliente/servidor na rede. Fan uso de servizos e os proporcionan. Nesta arquitectura por tanto non se poden agrupar os nodos e perde sentido falar de capas. Estas redes non resultan óptimas para todo tipo de servizos, en moitos casos por exemplo á hora de funcionar como servidor web, requiriría un custe moi alto para control da consistencia do sitio web, mantemento e configuración do servidor, etc... por contra, en situacións onde os nodos caen a miúdo, por exemplo un servidor atacado continuamente, a redundancia de nodos garante que o sistema siga funcionando. Outros servizos como o intercambio de arquivos, ou o procesamento compartido presentan máis vantaxes á hora de empregar unha arquitectura deste tipo como solución de implantación. Os modelos máis habituais son centralizados, puros ou descentralizados ou híbridos, segundo o peso de cada nodo individual na rede ou da existencia de servidores con responsabilidade de control e xestión no modelo. A adaptación deste modelo por parte dos ISP da lugar a P2P híbridas de servizo denominadas P4P (en inglés *Proactive network Provider Participation for P2P*). Outros modelos similares serían os P2M, que actúan en arquitecturas híbridas empregando o correo electrónico como soporte do envío de datos.

A xestión deste tipo de redes realízase fundamentalmente vía software. Neste caso o software deberá realizar as mesmas funcións xa vistas para a arquitectura de tres ou máis capas: frontal de comunicacións e xestión de transaccións. Por debaixo acostuman implementar servidores propios como Kademlia, eDonkey, Gnutella, FastTrack, BitTorrent ou OpenNap entre outros.

26.2.8. SERVIDORES.

26.2.8.1. Servidor web.

Trátase de servidores que provén o servizo WWW a través do protocolo HTTP. En esencia trátase dunha aplicación executándose nun servidor á espera de peticións HTTP por parte dun cliente respondendo cos documentos solicitados xeralmente páxinas web e os obxectos que enlazan: imaxes, arquivos de *script*, animacións, etc... En funcións máis avanzadas estes servidores engaden seguridade a través de conexións encriptadas con protocolos tipo HTTP Seguro ou HTTPS. Por regra xeral, os servidores de aplicacións intégranse en arquitecturas de mínimo tres **capas**:

1) **Primeira capa.** Capa de interacción cos usuarios, principalmente a través de navegadores web.
2) **Capa intermedia.** Capa dos servidores web, que poden estar distribuídos nun modelo de granxa de servidores. Cada servidor incorporaría os módulos necesarios para seguridade, linguaxes de servidor interpretados, correo, mensaxería, acceso a datos e outras funcionalidades.
3) **Terceira capa.** Capa de servidores de acceso a datos, como servidores de arquivos, base de datos ou informes.

Entre os **servidores web de uso máis estendido** actualmente atoparíanse:

- ✔ **Apache**. Un dos máis utilizados, por se un servidor libre que ofrece prestacións a nivel doutras solucións propietarias ademais dunha grande facilidade de uso e configuración.
- ✔ **Internet Information Server (IIS).** Servidor propietario con soporte para aplicacións .NET ou ASP entre outras.
- ✔ **Outros:** Java Web Server, AOLServer, Cherokee, Tomcat, lightHttpd, etc...

Os servidores web poden dispoñer de módulos para a execución de programas de servidor interpretados, como son os das tecnoloxías Python, PHP, ASP, JSP, Tcl, ...

26.2.8.2. Servidor de aplicacións.

Os servidores de aplicacións son servidores web con capacidade de procesamento ampliada, podendo executar aplicacións e compoñentes de lóxica de negocio e recursos relacionados como o

acceso a datos. Debido a isto permiten realizar o procesamento de aplicacións de cliente no propio servidor. Proporcionan soporte como *middleware* ou software de conectividade e para diferentes tecnoloxías de servidor. Por regra xeral, os servidores de aplicacións intégranse en arquitecturas de mínimo tres **capas**:

1) **Primeira capa.** Capa de interacción cos usuarios, principalmente a través de navegadores web.
2) **Capa intermedia.** Capa dos servidores de aplicacións, que poden estar distribuídos nun modelo de granxa de servidores. Un subconxunto dos servidores de aplicacións darán servizo aos usuarios/clientes mentres outro grupo encargarase de soportar a operativa común do dominio, como librarías ou aplicacións e servizos web dos que fagan uso as aplicacións para usuarios/clientes.
3) **Terceira capa.** Capa de servidores de acceso a datos, como servidores de arquivos, base de datos ou informes.

O servidor de aplicacións acostuma ter integrado un servidor web, para xestionar de maneira independente o servizo WWW a través do protocolo HTTP.
Ademais deste servizo presenta un amplo conxunto de **ferramentas**:

- ✔ Servidor web integrado.
- ✔ Contedor de programas de servidor (en inglés *servlets*).
- ✔ Contedores de obxectos de lóxica de negocio (por exemplo EJBs).
- ✔ Sistemas de mensaxería.
- ✔ Software de conectividade con bases de datos.
- ✔ Balanceo de carga.
- ✔ Xestión de límites e colas de conexións (en inglés *Pool*) para bases de datos e obxectos.
- ✔ Etc...

En esencia un servidor de aplicacións realiza as mesmas funcións que un servidor web, pero cando a demanda de uso é grande e estamos ante un sistema complexo a solución pasa por empregar un servidor de aplicacións que ofreza as seguintes **vantaxes**:

- ✔ **Centralización**. Centraliza nos servidores a administración e configuración da lóxica de negocio das aplicacións, de maneira que aspectos como o mantemento dos accesos a base de datos poden realizarse de xeito centralizado. Así mesmo cambios derivados de actualizacións, migracións ou recuperacións ante erros teñen lugar dende un único punto.

- ✔ **Seguridade.** Ao existir un único punto de acceso a datos pode reforzarse a defensa e os sistemas de control de erros nese punto, mellorando a súa xestión e protección.
- ✔ **Rendemento.** Como punto intermedio permite xestionar as peticións dos clientes á Base de datos.
- ✔ **Escalabilidade.** Un mesmo servidor de aplicacións pode dar servizo a varios clientes, e por tanto aumentando o número de servidores mellórase o rendemento do sistema.

Entre os **servidores de uso máis estendido** actualmente atoparíanse:

- ✔ **Jboss, Glassfish.** Servidores de aplicacións libres baixo licenza GPL.
- ✔ **BEA Weblogic, IBM Websphere, Oracle Application Server.** Alternativas propietarias integradas en paquetes de aplicacións con funcionalidades de xestión e monitorización estendidas.
- ✔ **Tomcat, Internet Information Server (IIS), Jetty**. Proporcionan funcións parciais de servidores de aplicacións, co cal en ocasións se definen máis ben como contedores de programas de servidor.

26.2.8.3. Servidor de acceso a datos.

Os servidores de acceso a datos ocuparían a última capa dos sistemas de información encargándose do acceso directo aos datos, existindo diferentes tipos segundo o sistema de información empregado para o seu almacenamento ou publicación:

- ✔ **Servidores de arquivos.** Neste tipo de servidores a información se almacena directamente en arquivos, por tanto a función destes equipos será a de permitir o acceso remoto aos mesmos dende os clientes ou outros servidores. Os protocolos máis habituais ofrecen servizo só dende redes locais pero en sistemas avanzados poden proporcionar servizos como FTP ou WebDAV para conexión remota a través de Internet. Actualmente o termo empregado para referirse a estes servidores é NAS (en inglés *Network-Attached Storage*), pero esta tan só sería a tecnoloxía máis habitual fronte a outras como DAS (en inglés *Direct Attached Storage*), baseada en SCSI ou SAN (en inglés *Storage Area Network*) baseada en fibra óptica. Non requiren un software moi específico coma outros tipos de servidores senón máis ben soporte para diferentes protocolos e tecnoloxías. Por norma xeral acostuman a estar dipostos en RAID (en inglés *Redundant Arrays of Independent Disks*), equipos de almacenamento redundante.

- ✔ **Servidores de bases de datos.** Albergan un ou máis sistemas de xestión de bases de datos (en inglés *database management system*, ou DBMS), software de xestión que se encarga da comunicación entre as aplicacións e as bases de datos. Permiten realizar operacións de definición, manipulación e seguridade dos datos a través dunha API de comunicación coas aplicacións e un linguaxe estruturado de consulta como o SQL. Permiten accesos simultáneos aos datos, seguridade e xestión de transaccións.

  Estes sistemas soen presentar ademais programas ou consolas de administración avanzadas para realizar as tarefas xerais de xestión da base de datos.
- ✔ **Servidores de informes.** Poden considerarse unha capa intermedia entre os servidores de datos e os de aplicación, onde se establecen servidores ou granxas de servidores que serven os datos en documentos predefinidos multiformato: follas de cálculo, PDF, XML, HTML, etc... O software deste tipo de servidores acostuma incorporar software de xestión para o servidor, e software de auto-edición de informes, para definir modelos de informes compostos de cabeceiras, imaxes, fórmulas, subinformes, etc... que se xerarán dinamicamente os informes a partir de consultas sobre os datos.

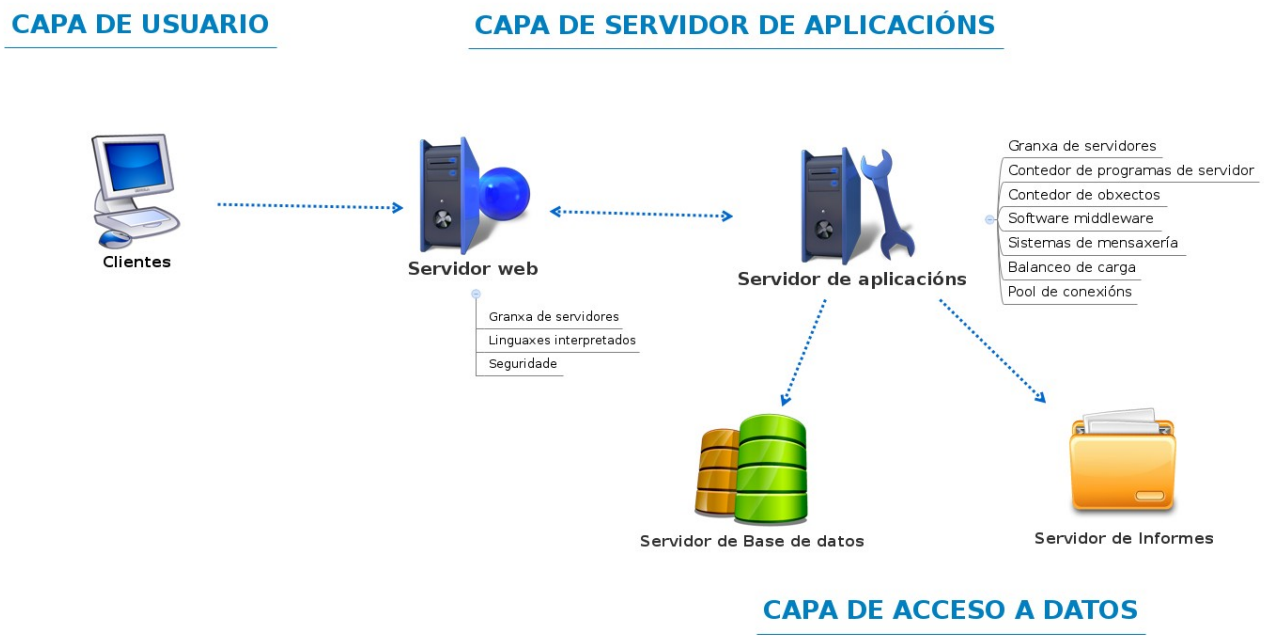


Figura 4: Arquitectura en 3 capas con servidor web, de aplicacións e base de datos.

Entre os **servidores de uso máis estendido** actualmente atoparíanse:

- ✔ **Servidores de arquivos.** Non requiren xestores especializados pero si soporte software aos protocolos: CIFS, NFS, SMB, FTP, WebDAV, etc... Así como utilidades tipo Samba ou FreeNAS.

- ✔ **Servidores de bases de datos.**
  - ✔ De licenza libre: PostgreSQL, MariaDB, Firebird, SQLite, Apache derby, ...
  - ✔ Dual, dependendo do seu 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 DO CLIENTE.

Os *scripts* do cliente son programas interpretados deseñados para executarse nos navegadores co obxectivo de dotar ás páxinas de maior interactividade co usuario e dinamismo nunha aproximación ás aplicacións de escritorio. O **funcionamento básico** dun *script* consiste en interpretar unha serie de comandos a través dos cales pode modificar e manipular obxectos e reaccionar ante eventos da interface como respostas a periféricos (rato, teclado, etc...), ou cambios nos elementos do documento (botóns, elementos de formularios, etc...). Os seus usos básicos son validacións, manipulación de formularios, procesamento de funcións e carga asíncrona de datos.

Os *scripts* de cliente proporcionan as seguintes **vantaxes**:

- ✔ Modificar o contido da páxina sen recargala do servidor en función das interaccións co usuario.
- ✔ Modificar parámetros de configuración do navegador e outros elementos da páxina web.
- ✔ Mellorar a interacción entre o usuario e o documento, en xeral a usabilidade.

Por contra, presentan un serie de inconvenientes ou **desvantaxes**:

- ✔ Problemas de accesibilidade, pois complícase a posibilidade de presentar alternativas a usuarios que non soporten a tecnoloxía de *script.*
- ✔ Problemas de seguridade, pois toda a lóxica de interacción aparece sen protección descargada no equipo do usuario, co cal dispón do código fonte do programa de *script*.

As **tecnoloxías de *script*** máis empregadas son Visual Basic Script, Javascript, coa súa evolución AJAX e PerlScript. O principal problema á hora de seleccionar unha tecnoloxía cando se deseña unha páxina web é o soporte que recibirá por parte dos navegadores, pois hai que lembrar que as linguaxes de *script* serán en última instancia interpretados no navegador.

a) **Javascript**. Baseado na linguaxe Java, é unha das linguaxes de script de uso máis estendido. É de sinalar, que ademais ten aplicación con outras tecnoloxías ademais da web coma en documentos PDF ou aplicacións de escritorio. Para permitir a interacción cos elementos dun documento web esta linguaxe dispón dunha API que implementa o DOM (en inglés *Document Object Model*) ou Modelo de Obxectos para a Representación do Documento, estandarizado polo W3C (en inglés *World Wide Web Consortium*). Nun intento por estandarizar esta linguaxe xorde o ECMAScript unha especificación da linguaxe aceptada como estándar ISO,

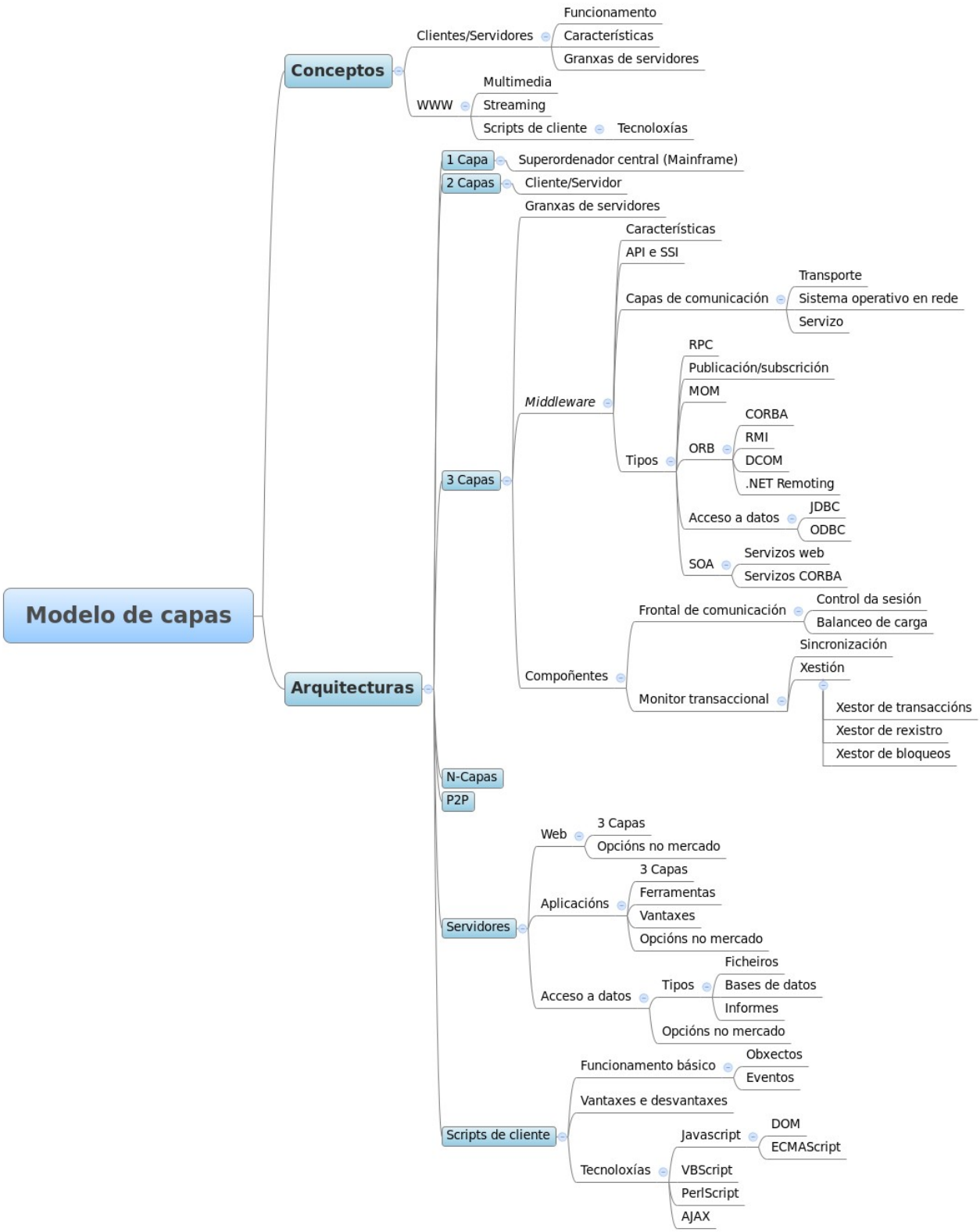
b) **Visual Basic Script.** Similar ao Javascript no tocante a funcionamento e estrutura pero baseado en Visual Basic. Ten menor soporte dentro dos diferentes navegadores, agás no Internet Explorer.

c) **PerlScript**. Baseado en linguaxe C o seu uso non está tan estendido coma as tecnoloxías anteriores aínda que ten menos limitacións. Debido a isto foi derivando cara linguaxe de servidor.

d) **AJAX**. Acrónimo de Javascript Asíncrono e XML (en inglés *Asynchronous Javascript And XML*). Esta é unha tecnoloxía de *script* de cliente asíncrona, de xeito que pode realizar cargas de datos sen que afecten á recarga da páxina. AJAX é un conxunto de tecnoloxías que fai uso de:

   1) XHTML e follas de estilo en fervenza (CSS) para a estrutura e deseño dos contidos.
   2) A linguaxe Javascript como linguaxe de programación para funcións e definición do programa cliente.
   3) O obxecto *XMLHttpRequest* para intercambio de información asíncrona co servidor. Por tanto, cómpre que o navegador soporte este obxecto, sendo empregado en ocasións o obxecto *Iframe*.
   4) XML e DOM como estándares asociados para intercambio de datos e manipulación do 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

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