Seguretat i qualitat en servidors web

Seguretat i qualitat en servidors web

Daniel Torrico Robledo
Héctor Alonso Martín
Marco Marín Martínez

PID_00177996

UOC
Universitat Oberta de Catalunya

Primera edició en llengua catalana: febrer 2012

Av. Tibidabo, 39-43, 08035 Barcelona

Disseny coberta: Natàlia Serrano
Imatge de coberta: Istockphoto

Realització editorial: Editorial UOC S.L

Daniel Torrico Robledo
Ingeniero Técnico de Telecomunicaciones con especialidad en Sistemas de Telecomunicaciones en la Universidad Carlos III de Madrid y experto en programación de aplicaciones web. Actualmente trabaja como Ingeniero de Sistemas en INDRA, realizando análisis de datos, automatización y optimización de procesos, arquitectura de sistemas, mantenimiento y gestión de proyectos de ampliación y mejora de sistemas para Telefónica de España.

Héctor Alonso Martín
Ingeniería en Informática de Sistemas por la URVITIL Foundation v3 Certified Administrador de Sistemas y BBDD / Programador 2001-2006 Administrador de Sistemas y BBDD 2006-2008 Arquitecto TI / Jefe de Proyectos 2008 (IBM GSE).

Marco Marín Martínez
Ingeniero técnico en informática de sistemas por la Facultad de informática de Barcelona (UPC), también es titulado por la UOC con el Master de Software libre de la UOC. Ha sido administrador de sistemas durante más de 7 años en especial en servidores UNIX (Linux, AIX, HP-UX, Solaris). En la actualidad ejerce como arquitecto de sistemas UNIX.

Índex

Capítol I

Tecnologia dels serveis web

Daniel Torrico Robledo

Comencem amb una pregunta simple: què són les tecnologies web? Per poder entendre bé l'assignatura, cal fer una mica de memòria històrica.

Podríem dir que la World Wide Web (www a partir d'ara) neix com a concepte en l'article «As We May Think» de Vannevar Bush el 1945, en què ja es proposava estendre la memòria humana utilitzant mitjans mecànics; tanmateix, parlar ara d'això potser sigui divagar molt sobre l'assumpte.

Basant-nos en fets més tangibles, Tim Berners-Lee, un informàtic del CERN (Organització Europea per a la Recerca Nuclear), és qui podríem considerar com el pare de la www l'any 1989. La idea de Tim era crear una eina de comunicació entre els científics nuclears del CERN per poder compartir informació utilitzant com a base la xarxa de xarxes que tot just acabava de néixer: Internet (o ARPANET per ser més exactes).

Per portar a terme el seu objectiu, va dissenyar un sistema d'hipertext en xarxa que permetia de manera gràfica i senzilla navegar a través d'enllaços a pàgines web o contenidors d'informació sense preocupar-se del format o de la localització geogràfica del contingut. Cada element o recurs s'identifica de manera global amb el seu URI (*Uniform Resource Identifiers*).

Ningú no podia preveure el que passaria després. Cada node que s'unia a la www creava nous fils en aquesta teranyina emergent de manera exponencial, cosa que va donar lloc a una complexa xarxa que avui en dia és un univers en si mateixa.

Aquest univers és el que tractarem en aquesta obra, concretament qui o què el forma, com i on s'emmagatzema tota aquesta informació, quins tipus d'informació existeixen i en què determina la seva naturalesa la forma com es presenta. És segur navegar per aquest univers? Quins mètodes de protecció existeixen? Totes aquestes i moltes més qüestions, les abordarem detalladament més endavant.

1.1. Els servidors de les tecnologies web

En aquest apartat coneixerem els tipus de servidors que podem trobar en la www de manera general. Quines són les seves funcions i per què els necessitem, quins tipus d'informació alberguen i quines tecnologies de programació poden implementar.

No obstant això i continuant amb el temari, hem de definir en primer lloc què és un servidor. Podem entendre com a servidor el conjunt de maquinari i programari que té la finalitat de proveir un determinat tipus d'informació o servei. En la www podem trobar servidors web, d'aplicació, multimèdia, de correu, *proxy*, de xat, ftp, Telnet, etc., però generalitzarem i els englobarem en tres grups, amb unes característiques molt diferenciades cadascun: web, aplicació i contingut.

A tall d'introducció, podem dir que el servidor web és bàsicament un programari que s'encarrega d'oferir una determinada informació que pot ser consultada o requerida des de qualsevol client compatible. Aquest programari se sustenta lògicament sobre un maquinari, però no caiguem en l'error d'identificar un servidor amb una màquina, ja que avui en dia és molt habitual parlar del «núvol» i de tecnologies distribuïdes, i això significaria que el nostre servidor s'estaria executant físicament com un tot en un maquinari geogràficament repartit pel globus en lloc de fer-ho només en una màquina concreta.

L'intercanvi d'informació entre client i servidor es realitza habitualment utilitzant el protocol HTTP en la capa OSI d'aplicació. El client acostuma a ser un navegador web que interpreta la informació que li ofereix el servidor i la presenta formatada, però aquest *modus operandi*, si bé és el genèric, té molts matisos que detallarem en l'apartat 1.2, com per exemple, qui executa accions?, el client?, el servidor?, tots dos?, en quins casos i quin tipus de continguts?

1.1.1. El servidor web estàndard

És la forma bàsica de servidor web que tenim. Simplificant arquitectures de maquinari i programari per explicar el concepte, podem entendre el servidor web estàndard com una màquina que subministra pàgines web a qui li ho sollicita. Aquestes pàgines web, com veurem més endavant, poden requerir informació ubicada en altres llocs, però, tornant a la senzillesa, imaginem que tot el *site* (o conjunt de pàgines web que conformen un servidor web) està

allotjat en un disc dur intern de la màquina on s'està executant el nostre servidor web.

Si el que volem és accedir a aquest lloc (*site*) concret, el primer que necessitarem és establir una nomenclatura unívoca per poder invocar-lo i així poder visualitzar-ne el contingut: necessitem la URL.

URL és l'acrònim d'*Universal Resource Locator* i va ser l'estàndard establert el 1994 per ubicar recursos a la xarxa. El seu format és molt similar a l'utilitzat en els sistemes de fitxers dels sistemes operatius, ja que té naturalesa jeràrquica d'arbre.

La seva sintaxi és la següent:

scheme://domain:port/path?query_string#fragment_id

Alguns valors són opcionals, com el *port* o el *fragment_id*, i si no s'especifiquen, prendran els valors per defecte.

Un exemple:

<http://www.uoc.edu>

Una vegada conegut l'URL, també hauríem de conèixer els conceptes d'adreça IP, domini, *hostname* i DNS.

- L'adreça IP és una etiqueta numèrica que identifica un element concret dins d'una xarxa IP.
- El domini és un nom que es dóna a un grup de recursos de la xarxa. El seu objectiu és fer més senzilla la identificació de recursos, ja que treballar directament amb l'adreça IP de cadascun faria molt complex i incòmode invocar cada recurs.
- *Hostname* és el nom d'un recurs concret dins d'un domini.
- DNS (*Domain Name System*) estableix una relació única entre un domini o *hostname* i la seva adreça IP.

Suposem que vull accedir al servidor web del meu exemple i que les seves dades són les següents:

- Adreça IP: 192.168.1.1
- Domini: exemple.es
- Hostname: 1

Per accedir a la màquina «1», que és la que està executant el servidor web, utilitzaré la seva URL de la manera següent:

<http://1.exemple.es>

Però, com ha pogut saber la xarxa IP que «1.exemple.es» és la meva màquina si ella només coneix adreces IP i no pas noms? Molt fàcil, el DNS li ha dit que l'adreça IP de «1.exemple.es» és «192.168.1.1». D'aquesta manera ens podem imaginar com ens resultaria de costós accedir a llocs sense DNS, ja que hauríem de conèixer l'adreça IP i recordar-la per a cadascun.

Una vegada carregada la pàgina web principal del lloc en el meu client (o explorador) web, puc anar navegant pels diferents continguts i fins i tot invocar-los directament escrivint la seva URL. Fins i tot es podria donar el cas que el mateix recurs al qual accedeixo mitjançant la seva URL en la meva màquina pugui ser invocat per una altra URL diferent. Com és possible? Doncs, és ben senzill: aquest recurs està publicat en dos dominis diferents. Per exemple i seguint amb el nostre escenari:

El nostre servidor web té com a domini <http://exemple.es> amb adreça IP 192.168.1.1 i adquireixo un altre domini que és <http://exemple2.es>. Només li he de dir al DNS que el domini «exemple2.es» també té l'adreça IP 192.168.1.1, i així podria accedir al mateix servidor o a un recurs d'aquest amb una URL diferent.

Aquest últim exemple hauria de servir per entendre que si bé cada servidor web té una única adreça IP que l'identifica, pot tenir múltiples dominis diferents que l'invoquin.

1.1.2. El servidor estès

Ja hem vist què és un servidor web estàndard en la seva forma més bàsica i coneixem els conceptes més comuns. Ara hi afegirem una mica de complexitat i presentarem alguns casos dignes de menció en aquesta assignatura.

Podem anomenar servidor web estès aquell que implementa funcionalitats complexes. Quan diem complexes ens referim a l'execució de codi, per exemple, per a la presentació de continguts dinàmics o tot el que se surt del llenguatge de text pla HTML purament dit.

Com a introducció a aquesta mena de servidors web de funcionalitats més avançades, podem començar veient què és CGI.

CGI o Common Gateway Interface és un estàndard o tecnologia que permet que un client web executi un programa en el servidor web. Aquesta definició s'ha de matisar, ja que no es pot executar qualsevol programa, sinó els que s'hagin creat en format CGI i en un llenguatge usualment de *script*. Com a exemples més representatius d'aquests llenguatges per programar CGI tenim Perl o Python.

Aquests CGI *scripts* poden ser invocats directament pel servidor web obrint un ventall de possibilitats molt interessants, com per exemple, passar-li determinades dades d'entrada i que l'*script* faci una acció concreta, generi una pàgina web nova o torni un resultat x perquè el servidor el presenti al client.

Un cop entès què és CGI i com pot aportar funcionalitat addicional al servidor web, estem en disposició de veure tecnologies encara més potents i flexibles en els nostres servidors web estesos: PHP i ASP.

PHP és una tecnologia de codi de servidor, a l'igual de CGI. Aquests conceptes els abordarem més detalladament en l'apartat 1.2, però per anar avançant coneixements podem dir que el seu codi s'executa en el servidor web i no en el client.

PHP s'implementa principalment en el contingut de web dinàmic i per accedir a bases de dades, ateses les seves característiques i la compatibilitat de comunicacions, a més a més el seu codi no pot ser vist pel client, ja que s'executa en el servidor, la qual cosa li confereix un alt grau de confiança i seguretat.

En ASP (Active Server Pages), bàsicament, hi trobem la mateixa potència i els mateixos avantatges que en PHP, però amb el matís que aquesta tecnologia pertanyent a Microsoft només es pot implementar en servidors ISS, cosa que la limita molt a l'hora de desenvolupar en entorns d'altres subministradors.

Una característica d'aquest llenguatge és la implementació dels ActiveX o, el que és el mateix, elements ja programats per utilitzar-los com a base o part d'un desenvolupament propi i funciona d'una manera similar als *applets* de Java.

1.1.3. Els servidors d'aplicacions

Un servidor d'aplicacions és un programari la finalitat del qual és proporcionar un entorn on es puguin executar aplicacions. Aquesta és una descripció molt general, ja que aquestes aplicacions, perquè es puguin executar, han de complir una sèrie de requisits i estar desenvolupades en determinats llenguatges, com veurem a continuació.

Hi ha diversos tipus de servidors d'aplicacions, però els basats en J2EE (Java Platform Enterprise Edition) i .NET són els més utilitzats. Com a clars exemples de cadascuna d'aquestes tecnologies i que podrem veure amb més freqüència en les grans empreses podem esmentar WebSphere (IBM) i WebLogic (Oracle) per a aplicacions que fan servir J2EE com a *core* i Internet Information Server (Microsoft) per a les dissenyades en l'entorn .NET.

En el servidor d'aplicacions s'estableixen els estàndards de desenvolupament i comunicacions perquè el desenvolupador no s'hagi de preocupar ni del maquinari ni del sistema operatiu que està per sota, la qual cosa aconsegueix homogeneïtzar i compatibilitzar els recursos existents de la xarxa independentment de la seva naturalesa. El servidor ofereix una API (Aplication Programming Interface) i un suport a diferents estàndards com XML, HTTP o SSL per establir les bases d'aquesta capa d'abstracció.

Els avantatges més ressenyables d'un servidor d'aplicacions, a part de la més evident comentada anteriorment (capa d'estandardització), són per exemple:

- Integritat del codi i configuració: el servidor controla de manera centralitzada l'aplicació, qualsevol modificació o actualització d'aquesta serà accessible per a tothom.
- Seguretat: el servidor controla els accessos a la informació.
- Rendiment: es pot parametritzar el servidor perquè les comunicacions client-servidor siguin les més òptimes per a cada entorn de xarxa.

Dit això, és raonable que sorgeixin dubtes de concepte, ja que amb el que hem vist fins ara un servidor web també pot executar codi o aplicacions (CGI, PHP, ASP…). Llavors, quines diferències hi ha entre un servidor web i un servidor d'aplicacions? La diferència radica bàsicament en el fet que el servidor d'aplicacions està dissenyat específicament per a aquesta finalitat, és més un *middleware*, és a dir, un intermediari que proporciona control i accés a les dades i les aplicacions. Té una estreta relació amb les bases de dades i els programes que executa no necessiten ser programats, ja que se serveixen com a mòduls al servidor web.

1.1.4. Els servidors de continguts

Un servidor de continguts, també anomenat CMS (Content Management System), és un programari destinat a facilitar i gestionar l'accés a una determina-

da informació o a determinades dades directament. Per entendre millor el concepte, vegem-ne un exemple.

En el cas anterior, el servidor d'aplicació s'encarregava de comunicar a l'aplicació X que estava executant amb les dades ubicades a la base de dades Y. Aquesta aplicació podia accedir a aquesta informació i la podia tractar o fins i tot mostrar al client, però el client mai no podria arribar directament a aquestes dades. No obstant això, el servidor de contingut ofereix informació directament al client a través de diferents estàndards.

Imagineu-vos, per exemple, un servidor de continguts multimèdia (XBMC) o un servidor FTP (Filezilla): el client es connecta al servidor i accedeix a pel·lícules, música o dades sense anar més lluny, de manera directa. És el servidor el que gestiona l'accés a aquestes dades i en garanteix la integritat.

Un dels seus principals avantatges està en el fet que qualsevol usuari amb permisos pot modificar o afegir contingut a un *site* sense tenir nocions de programació o maquetació, ja que el gestor de continguts és l'encarregat de generar dinàmicament tota la pàgina.

Dit d'una altra manera, els gestors de continguts són programari que s'executa en el servidor web gràcies als processadors de llenguatges interpretats o com a aplicacions suportades per un servidor d'aplicacions. Això significa que en una mateixa màquina podem estar executant el nostre servidor web i el nostre gestor de continguts. El primer és el que rep les peticions del client i el segon el que accedeix a la informació i la tracta per a tornar-la al servidor web dinàmicament usant PHP, per exemple, ja que hem vist la potència d'aquest llenguatge quant a l'accés a bases de dades.

Podem donar una volta més a la complexitat de l'arquitectura i fer diferents combinacions. Per exemple, imaginem que tenim a la mateixa màquina un servidor web i un d'aplicació i que una de les aplicacions que corren en aquest servidor d'aplicacions sigui un servidor de continguts multimèdia. S'entén que cadascun dels servidors té un paper molt concret i que entre ells es poden comunicar i complementar perfectament.

Arribats a aquest punt hauríem de tenir ja clars els tres tipus de servidors que podem trobar en entorns web, quines són les seves funcions i quan haurem d'usar cadascun d'una manera molt general. No obstant això, en els subapartats següents ens centrarem en els servidors web, ja que són l'objecte d'aquesta assignatura.

1.2. Serveis que s'ofereixen

En aquest capítol coneixerem els dos grans tipus de continguts web segons la seva naturalesa: estàtics i dinàmics.

Així mateix, aprendrem a classificar aquests continguts des del punt de vista d'on s'executen (client o servidor) utilitzant exemples de llenguatges de programació molt comuns en l'actualitat.

1.2.1. Publicació estàtica de continguts

Parlem de continguts estàtics des d'un punt de vista web quan el contingut no té variables, és a dir, el contingut sempre és el mateix i no canvia es consulti les vegades que es consulti.

Podeu imaginar una pàgina web bàsica, només de text, escrita en pur codi HTML amb etiquetes de format com un clar exemple d'aquest tipus de continguts.

Lògicament, els continguts estàtics sempre residiran en el servidor com a font origen de la informació, actuant com si es tractés d'una gran biblioteca plena de llibres. En aquest símil, el client seria l'alumne que va a la biblioteca, busca el llibre que vol consultar per un identificador unívoc i el llegeix, i qualsevol alumne que consulti aquest llibre trobarà sempre la mateixa informació escrita en ell.

No obstant això, si el que volem és crear un contingut web en què el mateix alumne busqui en una pàgina de la universitat si existeix aquest llibre i, si és així, que el reservi per tenir-lo quan ell arribi, és a dir, si volem que el nostre contingut web pugui interactuar amb el client i la informació que mostri sigui diferent en funció d'unes variables, llavors estarem parlant de contingut web dinàmic.

Tornant al que ens ocupa en aquesta secció i havent explicat els dos tipus de continguts web que podem trobar segons la seva naturalesa, introduirem un nou concepte per classificar aquest contingut des d'un altre prisma. El classificarem en altres dos grups diferents, però aquesta vegada, el contingut es diferenciarà per l'entitat que l'executa dins de l'arquitectura client-servidor: codi de servidor i codi de client.

Codi de servidor. Com es pot intuir sense gaire esforç, aquest codi és el que s'executa en el servidor exclusivament, és a dir, el codi que requereix un tractament previ en el servidor perquè una informació sigui llegible i intel·ligible pel

client i aquest la pugui presentar. Un exemple clar d'aquest tipus de codi és el llenguatge de programació web PHP.

En aquest llenguatge, el client fa una petició al servidor i li sol·licita informació concreta, una pàgina web o qualsevol altra dada. El servidor rep la petició i llança l'intèrpret PHP perquè executi el codi corresponent de manera dinàmica (*scripts* PHP). L'intèrpret torna el resultat d'aquestes execucions al servidor i aquest, al seu torn, el remet al client.

Cal aclarir que l'intèrpret no és res més que el mòdul lògic d'execució de *scripts* i que encara que en parlem com si es tractés d'una entitat lògica, en si mateixa diferent de l'entitat lògica servidor, forma part del bloc general del concepte servidor web i totes dues parts són indivisibles físicament, llevat que es recorri a binaris CGI independents, però això últim no ho abordarem en aquesta assignatura.

Codi de client: també serem capaços d'intuir que aquest codi és el que s'executa únicament en la part client de l'arquitectura. Podríem dir que és el llenguatge que no necessita un tractament previ del servidor, sinó que el mateix client és capaç d'entendre i presentar. Com a màxim exponent d'aquest tipus de codi tenim JavaScript.

La forma de treballar d'aquest llenguatge és força senzilla: el client fa una petició al servidor perquè li enviï una pàgina web, el servidor envia la pàgina web sense tractar, és a dir, text pla i etiquetes en què es defineixen els blocs de codi JavaScript. Aquesta pàgina «en cru» és llegida pel client, que la interpreta, la compila i l'executa directament per presentar les dades finals.

Amb aquests dos exemples de cada tipologia de continguts des del punt de vista client-servidor, es fa inevitable comparar els dos llenguatges per evidenciar les fortaleses i debilitats de cada filosofia de treball.

Avantatges de JavaScript sobre PHP:

- Molt bona llegibilitat del codi.
- JavaScript s'executa en el client, i per tant allibera el servidor de treball quan existeix una alta càrrega de transaccions en el servidor.

Desavantatges de JavaScript sobre PHP:

- El codi és visible per qualsevol usuari i, a més a més, s'ha de descarregar completament.

- No suporta programació orientada a objectes: classes i herència.
- No es pot connectar amb gestors de bases de dades: MySQL, Oracle, MS SQL Server, etc.

Hi ha moltes altres diferències menors, però realment aquestes diferències no són excloents, és a dir, no som davant un dilema en què ens vegem obligats a triar una manera de treball o una altra, sinó que són complementàries i aquest és, sens dubte, la més gran de les seves virtuts.

Podem implementar en el nostre contingut web pàgines amb codi PHP i JavaScript alhora per fer unes o altres accions, aprofitant el millor dels dos mons i creant un contingut òptim i responsable amb els recursos disponibles.

Recordeu que hem posat un exemple bàsic per entendre el concepte, però això no significa que JavaScript o PHP siguin llenguatges de contingut estàtic o dinàmic exclusivament, sinó que dependrà del que fem amb ells en classificar-los d'una manera o una altra:

Imagineu-vos que el codi de JavaScript que s'executa en el client és un «Hello world». El mateix si l'*script* de PHP imprimeix aquest text. El resultat és immutable, no canvia mai, no hi ha variables que el modifiquin. En els dos casos tenim un codi de servidor i un codi de client *estàtic*.

Però, i si el codi JavaScript no imprimeix un «Hello World» constant en la pàgina web, sinó que imprimeix el que jo li escrigui a la pantalla de manera dinàmica? I si fem el mateix en PHP? Tindríem un resultat variable en funció de la interacció del client, i per tant en tots dos casos el codi de servidor i el codi de client serien *dinàmics*.

1.2.2. Publicació dinàmica

Tal com es comentava en l'apartat anterior, el contingut web dinàmic és el que pot canviar en funció de determinats paràmetres.

En l'actualitat, aquest tipus de continguts és el predominant i és el que trobarem en pràcticament totes les pàgines web que visitem en una o altra forma, per la qual cosa conèixer-lo amb detall ens ajudarà a entendre millor la www.

Vegem ara uns exemples pràctics de llenguatges dinàmics per acabar de consolidar el concepte: Flash i AJAX.

Per ser puristes, Flash no és un llenguatge de programació pròpiament dit, és més aviat una plataforma de desenvolupament molt potent per crear entorns

animats i interactius. El llenguatge de programació que s'utilitza en aquest entorn és ActionScript que actualment va per la versió 3.0.

ActionScript és un llenguatge orientat a objectes el codi del qual, de la mateixa manera que JavaScript, s'executa en la banda del client, per la qual cosa, segons el que hem après en el punt 1.2.1, estaríem parlant de *codi de client*.

No obstant això, si bé gairebé tots els navegadors actuals suporten nativament codi JavaScript, en el cas de Flash no és així, per la qual cosa el client (o navegador web) necessitarà una petita ajuda addicional per poder «entendre» el codi.

Si volem que el client sigui capaç de representar codi ActionScript desenvolupat per a Flash, necessitarem que nostre navegador tingui correctament instal·lat un *plug-in* Flash. Aquest *plug-in* és un mòdul de la plataforma Flash que farà d'intèrpret, serà l'encarregat de compilar el codi i presentar-lo en el navegador. Vegem que ens ofereix aquesta plataforma i quins inconvenients té.

Avantatges:

- Alta qualitat visual: resultats realment bons per al maneig d'animacions i textos, i de gràfics vectorials.
- Animació i multimèdia: la millor tecnologia web existent avui en dia per incloure animacions, contingut interactiu, vídeo i àudio.
- Llenguatge de programació d'alt nivell orientat a objectes.

Inconvenients:

- Dependència d'un *plug-in* propietari per poder presentar les dades.
- Alt consum de recursos: en pàgines d'alt contingut Flash pot existir una experiència dolenta d'usuari si no es disposen de recursos HW suficients.

Un altre exemple de llenguatge de programació web dinàmic força representatiu avui dia seria AJAX (Asynchronous JavaScript And XML), tot i que, seguint la línia del cas anterior, tampoc no podem parlar d'AJAX com d'un llenguatge de programació en si mateix, sinó més aviat com d'un conjunt de llenguatges i eines.

Aquest mètode de programació web és realment innovador, ja que introdueix una nova entitat: una «capa» intermèdia entre el client i el servidor (anomenada

motor AJAX) que abstreu el client de l'arquitectura síncrona estàndard de tecnologies com, per exemple, les que hem tractat fins ara.

Aquesta nova manera d'entendre la interacció client-servidor es tradueix en grans avantatges:

- L'experiència d'usuari és molt més satisfactòria:
  - –No fa falta recarregar la pàgina en cada interacció.
  - –La funcionalitat de la pàgina no es bloqueja fins a rebre la resposta del servidor a causa del fet que les comunicacions es fan de manera asíncrona.
- Els temps d'espera es redueixen notablement, ja que les peticions i respostes al servidor són parcials. No s'envia la pàgina completa.
- El tràfic de xarxa es redueix igualment pels motius exposats anteriorment.
- No requereix *plug-ins* o *applets* com Flash o Java.

És cert que no tot són bondats i que aquesta tecnologia planteja determinats inconvenients:

- Falta d'integració amb el botó retrocedir del navegador. Es perd aquesta funcionalitat.
- Problemes de compatibilitat d'alguns navegadors amb XMLHttpRequest.
- Com que es genera de manera totalment dinàmica, no la podrem gravar a "Favorits".
- Com que es basa en JavaScript, podríem patir alentiments en cas que els recursos del client fossin molt limitats en pàgines amb un alt contingut AJAX.

Segons el que hem estudiat fins ara, podem afirmar que AJAX és una tecnologia de *codi de client*. Utilitzant llenguatge JavaScript de base juntament amb DOM, es potencia radicalment complementant-los amb la introducció de l'XML per establir comunicacions asíncrones amb el servidor.

En aquesta secció hem vist dues de les més conegudes tecnologies web dinàmiques de codi de client que hi ha i, sens dubte, les dues més utilitzades. Cadascuna té els seus punts forts i els seus punts febles, però cap és millor o pitjor que una altra, això dependrà exclusivament del problema que vulguem resoldre.

És per això que conèixer les característiques principals de les tecnologies més representatives ens ajudarà en el futur a prendre la millor decisió possible a l'hora d'haver d'enfrontar-nos a un repte de continguts web.

1.2.3. Serveis web

Fins ara, ens hem centrat a entendre els continguts web com a informació en pàgines web que es presenten en el client que la sol·licita de diferents formes: estàtica o dinàmica. S'executa en el servidor o en el client, i fins i tot s'estableix un intercanvi d'informació entre ells de manera bidireccional, però, com podríem establir una comunicació entre dues aplicacions totalment diferents a través de la www sense necessitat d'una interfície gràfica? Sembla fàcil, el dissenyador de l'aplicació A hauria de definir i acordar amb el dissenyador de l'aplicació B com s'entendran, però, i si els dissenyadors no es coneguessin o no arribessin a un acord o els llenguatges de programació no fossin compatibles entre si?

Aquestes i moltes altres qüestions obliguen a buscar un mètode estàndard per intercanviar informació o serveis entre aplicacions a través de la www: neix el servei web.

Podem definir *servei web* com un mètode de programari perquè dues aplicacions es comuniquin a través de la xarxa.

Els principals conceptes que hem de conèixer per entendre un servei web són:

- XML (eXtensible Markup Language): metallenguatge usat per descriure les dades que s'intercanviaran. Cal destacar que XML no és un llenguatge concret en si mateix, sinó que estableix un estàndard per definir llenguatges segons les necessitats (per exemple, XHTML seria un metallenguatge de HTML). Permet als desenvolupadors crear les seves pròpies etiquetes (o *tags*) per validar o interpretar dades.
- SOAP (Simple Object Access Protocol): protocol estàndard de missatgeria basat en XML encarregat de la transferència de les dades.
- WSDL (Web Services Description Language): és un format XML que s'utilitza per descriure un servei web, tant la seva estructura com el seu contingut.
- UDDI (Universal Description, Discovery and Integration): és el catàleg dels serveis web disponibles.

Tot això està molt bé, però com funciona un servei web realment? Passem a veure'l en detall:

Imaginem que hem creat una aplicació en C++ per a una empresa on es desen les dades personals dels empleats. Entre els requisits de desenvolupament es troba que es desenvolupi un servei web que accepti com a paràmetre d'entrada un DNI i torni totes les dades personals d'aquest empleat.

Per portar això a terme, a tall de resum, hauríem de fer el següent:

1. Desenvolupar el codi del programa que invocarà el servei web des de l'aplicació.
2. Crear un WSDL en què especifiqui quins camps viatgen en l'XML i què significa cadascun d'aquests camps. És a dir, crear l'especificació de la interfície de comunicacions, paràmetres d'entrada, de sortida i sintaxi de tots dos.
3. Publicar en l'UDDI una entrada amb el meu servei web i el seu WSDL. corresponent. Aquest pas, si bé és recomanat, no és obligatori, ja que si ja coneixem l'existència del servei web no fa falta buscar-lo, simplement invocar-lo i consumir-lo.

Una vegada realitzats aquests senzills passos, qualsevol aplicació que entengui de serveis web es podrà comunicar amb el nostre sistema, independentment del llenguatge de programació que hagi utilitzat.

És per això que el servei web implementa un estàndard de comunicacions abstraient-se del llenguatge, cosa que facilita la compatibilitat, escalabilitat i eficiència de les comunicacions entre aplicacions. Permet la comunicació i integració de processos i sistemes IT heterogenis.

Tornant al cas pràctic, imagineu-vos que l'aplicació de nòmines de recursos humans de l'empresa en què he desenvolupat la meva aplicació vol fer ús les dades que hi tinc emmagatzemades. La seva idea és calcular el plus d'antiguitat que correspon a cada empleat utilitzant la data d'alta en l'empresa. Doncs bé, si jo he definit un servei web que com a paràmetre d'entrada té el DNI de l'empleat i que com a paràmetres de sortida torna totes les seves dades personals, incloent-hi la data d'alta en l'empresa, entre altres, passaria el següent:

1. L'aplicació de recursos humans es connectarà via SOAP a l'UDDI per consultar els serveis web disponibles.

2. L'aplicació de recursos humans buscarà el meu servei web i quan el trobi li demanarà el WSDL associat que el descriu.
3. L'UDDI tornarà el WSDL corresponent al servei web que he publicat, el qual conté la descripció i l'estructura de l'XML d'invocació i de l'XML de resposta del servei web.
4. Amb tota aquesta informació, l'aplicació de recursos humans ja és capaç de construir una petició correcta. Utilitzarà el format que li ha especificat el WSDL per invocar-me amb el DNI dels empleats.
5. La meva aplicació rebrà la petició XML d'invocació amb el DNI, l'executarà i tornarà un XML amb tota la informació d'aquest empleat.
6. L'aplicació de recursos humans sabrà identificar gràcies al WSDL quin camp dins de l'XML és la data d'alta en l'empresa de l'empleat. L'extraurà i calcularà el plus d'antiguitat que li correspongui per ingressar-lo en la nòmina.

Aquest és, a tall de resum, l'esquema de comunicació més estès basat en el servei web que s'implementa, però no és l'únic.

1.3. Components bàsics i a l'entorn d'execució

Hem vist què és un servidor web, els diferents tipus de continguts i les tecnologies habituals, però, què es necessita perquè un servidor web pugui donar servei?

- Un mitjà a través del qual accedir-hi i a través del qual ell accedeixi als continguts: la xarxa.
- Una infraestructura física que el sostingui i es pugui executar: el maquinari.
- Un lloc especial on tenir aquest maquinari en condicions òptimes per assegurar-ne el funcionament correcte: el CPD.

1.3.1. La xarxa

Una xarxa, sens dubte, és un concepte molt general i abstracte, ja que hi ha xarxes en múltiples formes i tecnologies, però bàsicament i en el context d'aquesta assignatura, una xarxa informàtica és el conjunt d'elements de maquinari i de mitjans de comunicació que permeten l'intercanvi d'informació entre ells.

En capítols anteriors ja hem treballat amb molts dels conceptes que veurem en xarxes d'ordinadors o a la mateixa www: adreces IP, domini o DNS, per exemple, que es consideren ja coneguts a hores d'ara.

Centrem-nos en la forma més bàsica de xarxa per veure'l amb més claredat: una xarxa composta de dues computadores pròximes res més

Les xarxes d'àrea local o LAN són molt habituals avui en dia, pràcticament tots tenim una LAN a casa o a la feina. En el nostre exemple, podem usar l'estàndard Ethernet, ja que és el protocol més estès en l'actualitat per a aquest tipus de xarxes.

Tenim dues màquines (A i B) unides per un cable RJ45 a través d'un *hub* o un *switch*. Aquestes màquines, perquè puguin intercanviar informació, s'han de configurar perquè estiguin a la mateixa xarxa LAN o, altrament, encara que hi hagi un canal físic, no hi haurà un de lògic que les connecti. Aquesta configuració es determina amb una adreça IP, una submàscara de xarxa i una porta d'enllaç, si les dues màquines tenen els mateixos rangs i paràmetres anteriors, hi haurà comunicació.

Fins aquí tot bé, però, què passaria si hi hagués una sola xarxa LAN física i necessitéssim crear xarxes independents? I si tinc un tallafoc en la meva xarxa?

Introduirem dos conceptes de xarxa més complexos d'entendre, com són VLAN i DMZ, ja que són conceptes que haurem de manejar actualment en la majoria dels entorns web en què treballem.

VLAN (Virtual LAN): es tracta d'un mecanisme tremendament flexible, basat en l'etiquetatge, que permet comunicar computadores a nivell lògic com si estiguessin connectades al mateix concentrador de xarxa, però que poden estar físicament en segments de xarxa diferents. Aquesta tecnologia té grans avantatges. Podríem moure una computadora geogràficament i que mantingués la seva adreça IP i configuració de xarxa. Per portar això a terme necessitarem elements de xarxa de nivell 3 (per exemple, *switches*) si volem que siguin capaços d'entendre l'etiquetatge de les subxarxes virtuals.

DMZ (Demilitarized Zone): es tracta d'un tipus de xarxa orientada, sobretot, a proporcionar seguretat i estanquitat. El seu objectiu és aïllar de l'exterior una zona concreta de la xarxa interna i s'utilitza molt en xarxes que travessen un tallafocs. La DMZ s'usa per a fer visible a l'exterior part de la xarxa interna, però sense comprometre la seguretat de tota la xarxa interna.

Es pot accedir als equips ubicats en la DMZ des de la xarxa interna i externa del tallafoc, però aquests equips només poden accedir a la xarxa externa, no po-

den accedir a la interna. Aquest tipus de configuració de xarxa és actualment molt comú en les empreses i es denomina tallafocs de tres potes.

Ja que parlem de virtualització, hi ha altres arquitectures força interessants, com són els servidors virtuals. Un servidor físic, com un ordinador, pot tenir diversos servidors virtuals corrent alhora mitjançant virtualització i amb caràcter general, seria com tenir diverses màquines diferents alhora dins d'una de real, cadascuna amb la seva adreça IP diferent i totalment independents les unes de les altres.

En el punt 1.1.1 hem vist un exemple d'accés a una URL d'Internet, en què ha calgut comentar què és DNS i per a què serveix. Òbviament, en una xarxa LAN, DNS segueix sent indispensable per a identificar i correlacionar cada nom de màquina o *host* amb la seva adreça IP. També hem vist en aquest exemple com un servidor amb una IP podia tenir diversos dominis, cosa que també passa en les xarxes d'àrea local.

Hi ha diversos serveis auxiliars molt comuns i importants en una xarxa i que ens trobarem amb molta freqüència:

- FTP (File Transfer Protocol): és un protocol dissenyat especialment per a la transferència de fitxers entre computadores.
- SSH (Secure SHell): és un protocol que permet establir sessions segures entre computadores.
- VPN (Virtual Private Network): és un protocol de comunicacions que permet comunicar computadores com si estiguessin connectades a la mateixa xarxa LAN però a través de xarxes WAN.
- EMAIL: és un servei de xarxa que permet l'intercanvi de missatges entre computadores.

1.3.2. El maquinari

En aquest punt analitzarem què necessitem per oferir serveis web a nivell de maquinari. És evident que un servidor web es pot executar en pràcticament qualsevol computadora, des d'un *smartphone* fins a un servidor d'alt rendiment, però explicarem com afecta cada element del maquinari on s'executi el rendiment final del servei.

D'altra banda, és tan important assegurar un bon rendiment com assegurar la disponibilitat del servei, és a dir, que el servidor estigui donant servei en qualse-

vol circumstància. És aquí on estudiarem els mètodes més comuns d'alta disponibilitat.

Per portar a terme amb èxit el nostre disseny hem de tenir dos factors principalment en compte per a aquesta decisió: necessitats i pressupost.

- La CPU (Central Processing Unit): és l'encarregada d'executar les instruccions de la màquina. Com més gran sigui la seva velocitat de cicle, un nombre més alt d'instruccions podrà fer en menys temps, per la qual cosa aquest és un factor rellevant a l'hora de dissenyar un servidor web. Respecte al primer factor, hem de valorar què hem de manejar en el servidor: si el nostre contingut és dinàmic i s'executa en servidor (PHP, CGI…), necessitarem capacitat de còmput. Necessitem estimar i dimensionar el nombre de connexions simultànies esperades. És més, en el cas de moltes connexions simultànies, no ens valdria qualsevol CPU ràpida, sinó que, a més a més, hauríem de valorar el tipus d'arquitectura d'aquesta CPU (CISC o RISC). Es podria donar el cas que el nostre rendiment fos moltíssim més gran amb un processador Intel Itanium2 a 1,45GHz que amb un Intel Xeon a 3,5GHz a causa de les característiques de *cache* i de fils simultanis que tenen per cicle una i altra CPU, per la qual cosa cal insistir en tots els factors que hem de tenir en compte. No obstant això, serà el segon factor (pressupost) el que acabi per inclinar la balança d'un costat o un altre.
- Memòria RAM (Random Access Memory): la memòria RAM serà el contenidor de tota la informació que manegi el servidor o almenys ho hauria de ser si no volem grans penalitzacions per excessius accessos I/O a disc o fins i tot, *swaping* del SO. Valorant el factor u (necessitats), hem de tenir clar, igual que amb la CPU, quin serà el nostre contingut. Si estem muntant un servidor de continguts multimèdia, necessitarem una quantitat ingent de memòria RAM, però necessitarem molt poca si hem de muntar un servidor web estàndard. Aquest, de la mateixa manera que tots els altres, és un paràmetre que acabarà decidint el factor dos (pressupost). Hem de tenir en compte també que no solament hem de valorar la quantitat de memòria del sistema, sinó el tipus. Hi ha memòries ECC DDR3 i memòries SDRAM, i la seva velocitat i quantitat s'han de valorar separadament.
- Discos durs: aquest element de maquinari, encara que no ho sembli a priori, té una estreta relació amb l'anterior. En els discos durs s'emmagatzema tot el contingut de la màquina que es carrega a la RAM quan es necessita.

És molt més lent que la RAM, per això l'ideal és tenir el màxim nombre d'informació carregada a la RAM, però de vegades, per molts motius, això no és possible, la qual cosa ens obliga a buscar solucions en aquest element HW. Podem recórrer a discos en RAID0 o RAID5, fet que augmenta la nostra I/O, també a discos de 15k rpm, més veloços de l'habitual, fins i tot a discos SSD (Solid State Disk), els quals augmenten les prestacions d'una manera aclaparadora a causa de la seva naturalesa d'accés aleatori com la RAM. Tenim la possibilitat, fins i tot, d'emmagatzemar en cabines de fibra, però això ho veurem més endavant (apartat 1.3.3.1). Si tenim un servidor de continguts multimèdia amb Terabytes en pel·lícules és evident que no podem carregar en RAM aquesta informació, i per tant haurem de prestar especial atenció a aquest element, i, en aquest cas, es converteix en el més crític de tots sense cap dubte. La decisió dependrà, com sempre, dels factors u i dos.

- Targetes de xarxa: encara que no se li presti molta atenció a aquest element, les comunicacions són el factor més crític d'un servidor web: de res no em serveix una supermàquina si no puc arribar-hi. És especialment important triar targetes de xarxa d'alta qualitat i redundades, si bé la seva velocitat dependrà del tipus de continguts. De nou en el nostre exemple de servidor de continguts multimèdia necessitarem una amplada de banda enorme, i per tant podríem haver de recórrer al *trunking* de Gigabits Ethernet en casos extrems (adhesió de canals Gigabit Ethenet com un de sol). En cas de tenir un servidor web amb poc trànsit, amb una connexió de 10 Mbps n'hi hauria més que suficient.

Una vegada vistos els elements de maquinari més rellevants, ens podem centrar en com s'ha de fer per tal que el servei estigui sempre disponible o, almenys, que ho estigui la major part del temps possible. Hi ha diversos mètodes per a aconseguir tenir el servidor accessible i donant servei davant determinats esdeveniments d'error, però la casuística d'error és enorme: poden fallar elements de maquinari, de programari, de comunicacions... i cadascuna d'aquestes fallades es pot minimitzar d'una manera diferent de la resta. Us podeu imaginar, doncs, la possibilitat d'arquitectures a què poden donar lloc.

Passem a veure els tipus de redundància més habituals i necessaris per a servidors web:

- Redundància de maquinari: consisteix, almenys, a duplicar els elements de maquinari de la màquina perquè, en cas d'error d'un component, la màquina pugui seguir funcionant sense que el servei es vegi afectat. En servidors web, els elements susceptibles de ser objecte d'aquest tipus d'alta disponibilitat més habituals són la redundància de maquinari de discos durs, targetes de xarxa i fonts d'alimentació.
- Redundància geogràfica: aquest tipus de solució és molt utilitzada per a serveis crítics. Es basa a duplicar, com a mínim, les màquines que componen l'arquitectura i ubicar-les en llocs diferents i allunyats. El seu objectiu és evitar que un tall de comunicacions, d'energia o una catàstrofe natural que afecti una localització concreta, perjudiqui el servei, ja que l'altra localització seguiria assumint aquesta responsabilitat.
- Redundància de comunicacions: com dèiem anteriorment, les comunicacions exerceixen un paper fonamental en un servidor web. En serveis crítics es fa necessari, almenys, duplicar totes les línies de comunicació, i tots els elements de la xarxa (*switches*, commutadors, tallafocs…) per garantir, com a mínim, dos camins totalment independents per arribar al nostre servidor.
- Redundància elèctrica: de la mateixa manera que en el cas anterior, es fa necessari assegurar el subministrament elèctric als nostres servidors. Això s'aconsegueix amb connexions d'energia redundants i independents per a cada màquina.

Una vegada redundada la nostra arquitectura física, necessitem introduir elements que minimitzin els possibles punts d'error i garanteixin el servei encara més. Tingueu en compte que si una màquina A em cau, tindria la màquina B per donar servei, però en el temps que trigo a engegar-la i posar-la com a principal, no estic donant servei i no he aconseguit garantir l'alta disponibilitat malgrat tots els esforços que hem fet en els punts anteriors. Llavors, què podem fer?

Per resoldre aquesta qüestió, hi ha diverses solucions d'arquitectura, com, per exemple, el *clustering* o el balanceig de càrrega.

En el cas anterior, si les meves màquines formen part d'un clúster actiu, la caiguda d'una no impactaria en el servei. De la mateixa manera, si tingués les dues màquines en actiu amb un balancejador de càrrega davant, tampoc no ho faria, però estem introduint un element més i cada element extra genera un nou punt d'error, que és el que intentem evitar, encara que això es pot minimitzar redundant el balancejador.

La nostra arquitectura serà tan segura i estable com el seu component més feble.

1.3.3. El CPD

Com hem tractat en el punt 1.3.2, les comunicacions i l'energia dels nostres servidors són clau per assegurar la nostra arquitectura davant possibles fallades i pèrdues de servei, però, quant costaria crear tots aquests serveis per nosaltres mateixos? Imagineu-vos haver de construir una minisala en cada lloc de la nostra arquitectura redundada geogràficament, comunicar aquestes sales de manera redundant, dotar-les de diversos subministraments elèctrics independents, etc.

Evidentment, el cost del projecte el faria pràcticament inviable i és en aquest marc on apareix el concepte que aprendrem i coneixerem: el CPD.

El CPD, o Centre de Procés de Dades, és un edifici o sala dedicat exclusivament a proporcionar les condicions òptimes per al funcionament d'equips electrònics, garantir-ne la seguretat i dotar-los de serveis de subministrament i manteniment.

Existeix una normativa internacional en què s'especifiquen les característiques que ha de complir un CPD per complir l'estàndard.

Podem esmentar com a exemple les següents:

- La temperatura, que ha de ser de 22,3 ºC a la sala freda on s'ubiquen els servidors.
- La seguretat, ja que només pot accedir a la sala personal autoritzat. S'ubiquen càmeres infraroges cobrint tots els angles, accés mitjançant portes amb confinament i detectors de moviment, entre altres mesures.
- L'energia elèctrica, ja que es necessita doble connexió de servei independent, grups electrògens autònoms i SAI.
- Diversos canals de comunicacions i redundància d'electrònica de xarxa.
- Sistemes contra incendi i sísmics avançats.
- Moll de càrrega i magatzem, ja que gairebé tot l'equipament entra i surt en camions.

1.3.3.1. L'emmagatzematge compartit

Perquè el CPD garanteixi la integritat de les dades i la seva seguretat, a més a més de dotar-los d'alta disponibilitat, es dissenya una xarxa exclusiva i independent per a l'emmagatzematge de les dades.

Com hem dit, el cost d'implementar redundància en les arquitectures és molt alt i inviable en la majoria dels casos; és per això que en el CPD es crea aquesta xarxa comuna perquè els servidors que ho requereixin puguin accedir a aquestes

infraestructures avançades, i reduir dràsticament els costos d'accés a aquesta tecnologia. Les xarxes d 'emmagatzematge més usades en l'actualitat són SAN (Storage Area Network) i NAS (Network Attached Storage).

En la SAN es té com a objectiu comunicar de manera ràpida, fiable i segura els servidors i els elements d'emmagatzematge remot compartit com, per exemple, *arrays* de discos o cabines. Com a norma general, s'utilitza tecnologia *fiber channel* per a aquesta xarxa, la qual veurem més endavant a causa del seu rendiment, encara que en l'actualitat, està adquirint un paper rellevant la tecnologia iSCSI, que també comentarem per l'estalvi en costos que representa; l'elecció de l'una o de l'altra dependrà com sempre del balanç entre necessitats i pressupost que hem comentat en apartats anteriors.

Entrant en detall, una xarxa d'emmagatzematge SAN es diferencia de la resta, bàsicament, per la forma d'accés de baix nivell, és a dir, en lloc d'accedir a un fitxer X, com si d'un sistema de fitxers es tractés, tal com fan les xarxes d'emmagatzematge NAS que veurem més endavant, ho fa a un bloc o adreça de disc. Aquests «discos» no són discos reals, es crea una entitat anomenada LUN, que és un conjunt de blocs de disc físic. Aquesta LUN o disc virtual són els elements base sobre els quals es consulten les dades i es transfereixen per la xarxa.

En la SAN es poden diferenciar tres capes:

- Capa de *hosts*: en aquesta capa s'ubiquen els elements extrem A de la xarxa maquinari i programari, com ara els servidors i els seus dispositius d'interconnexió a la xarxa (HBA).
- Capa de comunicacions: en aquesta capa trobem tot el cablejat de fibra òptica (o coure, ja que el protocol és compatible amb el medi elèctric), i els *hubs* i *switches* d'interconnexió de la SAN, que serien els elements intermedis.
- Capa d'emmagatzematge: en aquesta capa s'ubiquen els elements extrem B de la xarxa maquinari i programari, com ara els *arrays* de discos i cabines de cintes on es desen les dades.

També podem distingir la SAN per tipus de connexió:

- Fiber channel
- iSCSI

Tecnologia fiber channel: aquest tipus de tecnologia té com a base el protocol SCSI, que s'utilitza per a la comunicació de discos de manera local. El princi-

pal problema de SCSI és que les interfícies i els mitjans de transmissió deixen de ser efectius amb una distància de metres, la qual cosa fa inviable crear una xarxa d'aquest tipus. Amb aquest objectiu, neix fiber channnel, per comunicar elements SCSI a través de fibra òptica, eliminant el problema de la distància i fent viable les xarxes a través d'aquest protocol.

Fiber channel pot assolir velocitats de fins a 8 Gbps i, com que té una xarxa de fibra òptica exclusiva i independent, tot el trànsit d'emmagatzematge es cursa sense interferències de xarxa, de manera fiable, ràpida i segura. Per contra, crear una xarxa d'aquest tipus i dotar de targetes de fiber channel (o HBA) els servidors encareix moltíssim la solució.

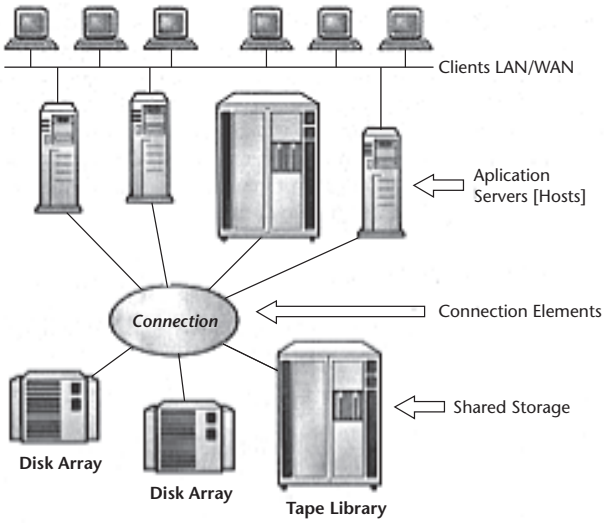
Hi ha tres tipologies:

- Punt a punt (FC–P2P): connexions *end to end* directes.
- Anell arbitrat (FC–AL): els elements s'uneixen dintre seu formant un anell. En afegir-hi un nou element o suprimir-lo, s'interromp l'anell.
- Medi commutat (FC–SW): la més comuna de les tres. Tots els elements s'uneixen a la xarxa a través de *switches* que optimitzen les connexions.

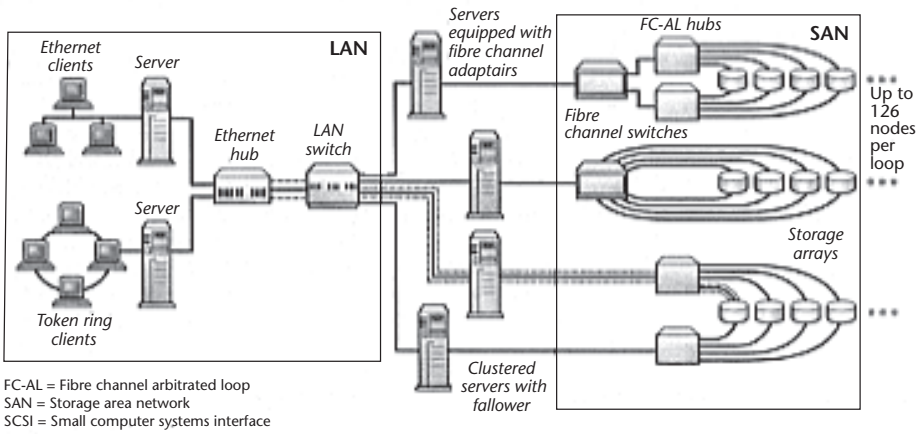
Tecnologia iSCSI: Aquest tipus de tecnologia té com a base el protocol SCSI igual que l'anterior, però en aquest cas, per eliminar el problema de la distància no s'utilitza una cara xarxa de fibra òptica, sinó que es recorre a les xarxes TCP/IP per transportar el protocol. A causa de l'avanç de les xarxes Ethernet actuals, la velocitat estàndard de les quals se situa en 1 Gbps, és molt menys costós implementar una xarxa d'emmagatzematge SAN dedicada amb cablejat, *switches* i targetes Ethernet que fer-la de fibra òptica. Aquest fet està inclinant la balança cap a aquest tipus de solucions en les arquitectures que no necessitin tant amplada de banda com fiber channel, ja que aporten els mateixos avantatges a un preu molt menor.

Com hem comentat al principi de l'apartat, hi ha un altre tipus de xarxa d'emmagatzematge que competeix directament amb SAN, si bé no són excloents i es poden combinar, cadascuna té els seus avantatges i desavantatges. Aquesta xarxa s'anomena NAS.

En una xarxa NAS, l'emmagatzematge és local al sistema de fitxers, per la qual cosa els equips realitzen les peticions de dades i intercanvien informació als sistemes de fitxers de manera remota mitjançant protocols CIFS i NFS. Recordeu que en SAN, les dades es demanen directament al sistema de fitxers, com hem vist en la definició de LUN.



Example of a Simple SAN Configuration



Example of a Complex SAN Configuration

Avantatges de NAS sobre SAN:

- Capacitat de compartir unitats d'emmagatzematge.
- Menor cost de la tecnologia i elements de xarxa.
- Comparteix la infraestructura de xarxa Ethernet.
- Gestió molt més senzilla.

Desavantatges de NAS sobre SAN:

- Menor rendiment.
- Menor seguretat a causa de l'ús compartit de les comunicacions.
- Menor escalabilitat.

Hem comentat que NAS utilitza protocols per accedir al sistema de fitxers de manera remota, vegem-los breument:

- NFS (Network File System).
- CIFS (Common Internet File System).

NFS: aquest protocol de nivell d'aplicació OSI que permet que diferents sistemes connectats a una mateixa xarxa accedeixin a arxius remots com si es tractés d'arxius locals.

CIFS: és un protocol de nivell d'aplicació OSI que permet compartir arxius i impressores (entre altres coses) entre nodes d'una xarxa. És utilitzat principalment en ordinadors amb sistemes operatius DOS i Windows. Existeix una adaptació del protocol per a Linux anomenat SAMBA.

Enllaçant amb els punts anteriors d'alta disponibilitat i CPD, convé saber que en les arquitectures més habituals, les cabines de discos se solen redundar geogràficament i s'interconnecten per la xarxa d'emmagatzematge. Això assegura que la informació emmagatzemada sempre estigui disponible per als servidors que les utilitzen davant de qualsevol contingència que pugui ocórrer, cosa que fa els dissenys molt més robustos.

1.3.3.2. Els sistemes de còpia de seguretat

Si bé hem intentat minimitzar els punts d'error i d'afectació de servei, no els podem eliminar completament, i això ens obliga a tenir sistemes de còpia de seguretat i *backup* que ens permetin recuperar el servei davant un problema eventual.

Una còpia de seguretat és almenys una duplicació de dades de les que es volen assegurar.

Els principals problemes que ens trobarem a l'hora de dimensionar i gestionar còpies de seguretat són:

- Espai d'emmagatzematge necessari.
- Impacte en el rendiment dels sistemes.
- Cost del maquinari i del programari de la infraestructura de *backup*.
- Temps de recuperació.

Hi ha múltiples mitjans en els quals fer còpies de seguretat, com ara unitats òptiques o discos, però els més usuals són els robots de cintes i les cabines de discos.

De la mateixa manera, tenim diversos tipus de còpies de seguretat, per exemple:

Atenent l'estat de les dades, els tipus de còpia de seguretat més habituals són:

- Còpies en fred: les que es fan amb la informació origen bloquejada, és a dir, no s'hi permet l'accés mentre es fa la còpia.
- Còpies en calent: les que es poden fer sense haver de bloquejar la informació origen bloquejada. S'hi permet l'accés mentre es fa la còpia.

Atenent el contingut de les dades, els tipus de còpia de seguretat més habituals són:

- Completes: es copia tot el contingut origen.
- Diferencials: es copia només allò que existeix en origen que no existeixi en l'última còpia de seguretat realitzada, és a dir, només els canvis des de l'última vegada que es va fer un *backup*.

Tots aquests temes i molts altres que hi estan relacionats els tractarem en profunditat més endavant, en el capítol corresponent.

1.3.4. Les bases de dades

Una base de dades és una col·lecció d'informació que està organitzada de tal manera que sigui fàcilment accessible, administrable i actualitzable.

Per poder accedir a una base de dades cal un sistema d'accés i gestió, anomenat *gestor* (SGBD, Sistema Gestor de Base de Dades), que ens permetrà recuperar, accedir i organitzar la informació.

Segons el model de dades que implementi el SGBD, podem classificar les bases de dades en dos tipus, relacionals i no relacionals; cadascun posseeix característiques que poden ser òptimes en diferents entorns d'explotació, segons la finalitat, el tipus i la complexitat de les dades que volem emmagatzemar.

Les característiques principals del model de dades relacional són les següents:

- Estructura les dades en forma de relacions que es modelen mitjançant taules. L'estructura denominada «relació» permet representar tant els objectes com les relacions entre aquests objectes.
- Permet la incorporació d'aspectes semàntics de l'univers del discurs mitjançant l'establiment de regles d'integritat. Aquestes regles permeten traslladar a l'esquema conceptual restriccions o comportaments de les dades presents en l'univers del discurs que no es podrien modelar exclusivament amb taules.
- Està basat en un model matemàtic, amb regles i algoritmes algebraics establerts, fet que permet el desenvolupament de llenguatges d'accés i manipulació potents.

La gran majoria de bases de dades usades actualment són relacionals i són utilitzades en les més diverses aplicacions, des de comerç electrònic, bancs de dades mèdiques, facturació, recursos humans i pràcticament qualsevol aplicació.

Però no totes les bases de dades modernes són relacionals, hi ha molts casos en què altres paràmetres, com la velocitat d'accés a les dades, són més crítics.

Hashing: en molts casos, l'accés a dades és de només lectura, perquè no es preveu que les dades siguin modificades o perquè aquesta modificació es realitza de manera programada mitjançant processament per lots sobre una gran quantitat de dades i de manera no freqüent. En aquests casos és possible usar estructures de dades que permetin un accés molt ràpid.

En el *hashing*, la clau que serà buscada es transforma, mitjançant un algoritme, en un número anomenat *hash*. Les dades s'organitzen de tal manera que el *hash* és el número de registre on s'emmagatzema el valor de la clau; d'aquesta manera, per a grans conjunts de dades, el *hashing* és molt més ràpid que buscar un índex perquè les dades es recuperen en molts menys accessos al disc. En la

pràctica, diverses claus poden donar lloc a un mateix *hash*. Aquests casos es coneixen com a *col·lisions* i, en la mesura que es pugui, s'han d'evitar.

Des del punt de vista de l'arquitectura, podem classificar els SGBD en dos tipus: centralitzats i distribuïts.

Un sistema de base de dades centralitzat és el que s'executa en un únic sistema computacional sense, per a aquest efecte, haver d'interactuar amb altres ordinadors. El rang d'aquests sistemes comprèn des dels sistemes de bases de dades monousuari que s'executen en ordinadors personals fins als sistemes de bases de dades que s'executen en sistemes d'alt rendiment. Normalment, els sistemes de base de dades monousuari no solen proporcionar moltes de les facilitats que ofereixen els sistemes multiusuari. En particular, no tenen control de concurrència i tenen sistemes de recuperació precaris o inexistents.

En un SGBD distribuït, la base de dades s'emmagatzema en diversos equips que es poden comunicar al seu torn per diferents mitjans (des de xarxes d'alta velocitat fins a línies telefòniques). No comparteixen memòria ni discos i les seves mides poden variar tant com les seves funcions, i poden comprendre des d'un únic equip fins a grans sistemes. Es denomina amb el terme d'*emplaçaments* o *nodes* tots les equips que pertanyen a un sistema distribuït.

Les bases de dades distribuïdes es troben normalment en diversos llocs geogràfics diferents, s'administren de manera separada i posseeixen una interconnexió més lenta. En un sistema distribuït es donen dos tipus de transaccions, les locals i les globals. Una transacció local és la que accedeix a les dades de l'únic emplaçament en què s'iniciarà la transacció. D'altra banda, una transacció global és aquella que, o bé accedeix a les dades situades en un emplaçament diferent d'aquell en què es va iniciar la transacció, o bé accedeix a dades de diversos emplaçaments diferents.

Per poder desenvolupar aquesta arquitectura es necessita un client intelligent que pugui sol·licitar serveis d'un servidor en xarxa. En el cas dels servidors web, hi ha una capa de mediació, normalment implementada en llenguatges de programació web, que processen les peticions de l'usuari i, mitjançant crides a procediments, sol·liciten dades al servidor, cosa que permet explotar la base de dades.

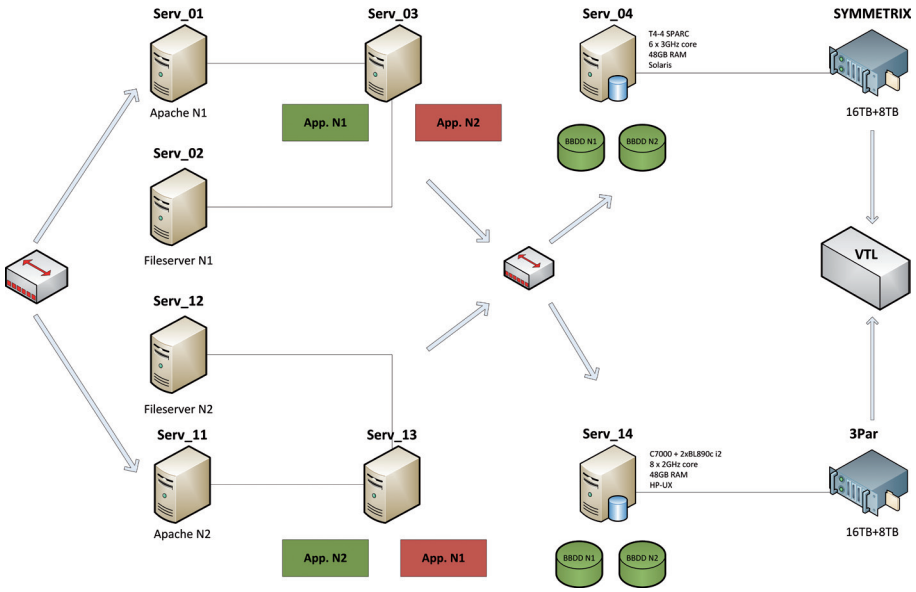
Més endavant veurem els mètodes habituals de connexió a les bases de dades, i les solucions tecnològiques disponibles en el mercat, solucions d'alta disponibilitat i redundància.

1.4. Infraestructura del servidor web

Quan parlem d'infraestructura d'un servidor web ens referim a tots els elements físics que intervenen directament en la provisió del servei o la seva disponibilitat.

Entre els elements de maquinari hi ha les màquines, tant si serveixen com a servidors d'aplicació, de bases de dades o *front*-ends, que estan localitzades en emplaçaments en condicions d'energia, espai i climatització òptimes, i tots els dispositius de connectivitat, *routers*, *switches*, concentradors, balancejadores, unitats d'emmagatzematge, etc.

Partim d'un exemple d'infraestructura:



Infraestructura per a un sistema de dos negocis, amb solucions de* backup*, alta disponibilitat i redundància geogràfica

Analitzem el sistema plantejat per parts. Primer, el sistema dóna servei a dos negocis que poden estar relacionats, per exemple, un que s'ocupa de les comandes de clients i un altre de les comandes a proveïdors.

Veiem que la infraestructura del sistema està repartida en dos CPD, o centres de dades. Això proveeix al sistema complet una solució de redundància geogràfica molt útil en casos de talls prolongats d'energia o indisponibilitat de la xarxa.

El primer element del sistema és un balancejador de càrrega, capaç de cursar peticions cap a una i/o altra línia.

A continuació, tenim els *front-ends*, servidors de presentació, encarregats de rebre les peticions dels usuaris i cursar-les cap als servidors d'aplicació per tornar la resposta a l'usuari en forma de pàgines web. La potència d'aquests equips dependrà de la densitat de peticions que hagin d'atendre, i poden ser des d'ordinadors personals fins a equips de 8 o 10 cores en casos d 'alta càrrega.

Els servidors de presentació han de contenir el programari apropiat per a la seva finalitat. Els casos més habituals són servidors Apache/Tomcat (multiplataforma de codi obert), IIS (implementat per a Windows), NGINX (multiplataforma i de codi obert).

És important decidir el programari que es vol triar. Moltes de les solucions disponibles requereixen llicències costoses i que després representen una gran part del cost de manteniment.

L'element següent és el servidor d'aplicacions, el cor del sistema, encarregat d'executar la lògica del negoci i, habitualment, una part crítica del sistema. És per això que per a negocis d'alta càrrega de transaccions són habituals equips de gran capacitat de càlcul i molta memòria RAM. Alguns exemples d'equips són la sèrie HP Integrity rx (fins a 16 processadors dual-core i 512 GB de RAM en el cas del rx8640), la sèrie T4 d'Oracle (dos processadors quad-core i 512 GB de RAM en el cas del T4-2) o la sèrie Power Express d'IBM (fins a 16 cores i 512 GB de RAM en el cas del Power 740 Express).

En aquest punt cal aclarir que no tots els servidors valdran per al nostre propòsit. Alguns implementen arquitectures Intel Itanium, SPARC, PA-RISC o Intel x86, i haurem de triar l'arquitectura física que més s'adapti a la nostra aplicació, en cas que ja estigui desenvolupada, o que més flexibilitat ens permeti en el futur.

En la figura veiem que cada servidor d'aplicació executa una instància activa d'un negoci (App N1, App N2, en verd) i una instància inactiva d'un altre. Aquesta configuració, molt habitual en els grans servidors, serveix com a solució de disponibilitat en cas d'error d'algun dels servidors d'aplicacions, i evita una pèrdua prolongada del servei. Aquesta solució s'anomena clúster actiu-passiu i la veurem més endavant amb deteniment. Les aplicacions s'executaran sobre plataformes de desplegament, com per exemple IBM WebSphere o Oracle WebLogic.

Seguim amb els servidors de base de dades, que anomenarem *back-end*, que, a través d'un balancejador, cursen les peticions d'aplicació cap al conjunt de da-

des. Aquests servidors solen suportar grans càrregues de transaccions, per la qual cosa el seu rendiment és d'importància crítica. Entre les solucions més esteses podem esmentar *enclosures* (és el cas del HP-C7000), que contenen unitats de processament (*blades*), com el BL890c i2 de fins a 32 cores i 1,5 TB de RAM, que se «sumen» per presentar-se com un únic recurs, o servidors com l'Oracle T4-4 de 32 cores i 1TB de RAM, optimitzats per suportar motors de base de dades.

Finalment, presentem les solucions d'emmagatzematge i *backup*, a través de dues cabines, una EMC-SYMMETRIX (capaç d'emmagatzemar fins a 2,06 PB en cas del model SYMMETRIX-VMAX) i una altra HP-3Par (pot emmagatzemar fins a 1,6 PB en el cas del P10000), que són les solucions més esteses en grans aplicacions corporatives, sincronitzades per mantenir una redundància de dades que previngui algun desastre en cas de fallades de discos o dades corruptes.

Evidentment, l'exemple proposat es correspon amb una aplicació corporativa en què els requisits de rendiment, disponibilitat i escalabilitat són elevats. No tots els sistemes es resolen amb infraestructures tan complexes, ni cares. I hi ha solucions disponibles, com la virtualització o el *cloud computing*, que faciliten en gran manera els considerants d'un disseny, les solucions esmentades i altres que hem comentat en aquest apartat i que es veuran amb més detall més en davant.

1.4.1. Infraestructura pròpia o llogada

Una vegada definida la infraestructura necessària, cal implementar-la. No hi ha dubte que aquesta implementació comporta una important despesa econòmica, que variarà segons les solucions tecnològiques elegides. Aquesta despesa, coneguda com a cost total d'adquisició, representa la despesa més important a l'hora de posar en funcionament el servidor.

Es defineix com a *retorn de la inversió* el temps necessari per recuperar la inversió inicial i representa un paràmetre molt important a l'hora d'avaluar la viabilitat econòmica del projecte. El temps de retorn ens dóna una idea cabdal dels beneficis estimats del negoci.

En aquest context se'ns presenten dues solucions: una, la més evident, consisteix en l'adquisició de tota la infraestructura necessària, i dos, el lloguer d'infraestructura. Hi ha múltiples proveïdors dedicats al lloguer d'infraestructura, que atenent el benefici que representa el factor d'escala, posen a disposició la infraestructura física com a servei.

L'elecció de la solució òptima dependrà de diversos factors, com ara:

- Inversió inicial: l'adquisició de tota la infraestructura implica una inversió més gran i, per tant, una barrera d'entrada que cal considerar.
- Flexibilitat: el lloguer d'infraestructura ens aporta més flexibilitat a l'hora de fer canvis, sempre que el proveïdor pugui satisfer les noves necessitats.
- Costos de manteniment: en el lloguer d'infraestructura gran part dels problemes associats al manteniment i a la gestió de la màquina i plataformes es deixen en mans del proveïdor.
- Escalabilitat: les solucions llogades ja estan preparades per a satisfer més demandes de capacitat de processament, emmagatzematge i amplada de banda.
- Control: l'adquisició d'infraestructura ens permet tenir més control sobre el sistema.
- Disponibilitat: si la infraestructura és pròpia, cal dotar el sistema de solucions tecnològiques que assegurin la disponibilitat. En cas de llogar la infraestructura, aquesta responsabilitat es trasllada al proveïdor a través d'acords de nivell de servei (ANS) reflectits clarament en el contracte.

Una opció o una altra dependrà tant de la viabilitat econòmica com de la planificació i estimació de creixement del negoci a curt/mitjà termini. El lloguer d'infraestructura és l'elecció més barata, i l'adquisició d'infraestructura pròpia es pot considerar sobre la base d'estimacions a llarg termini que és quan es preveu que la inversió serà amortitzada.

1.4.2. Els models clàssics: hosting/housing

Com s'ha comentat anteriorment, la inversió inicial representa un dels obstacles més grans a l'hora de posar en funcionament el servidor. De fet, és possible que l'adquisició d'infraestructura pròpia sigui inviable per al nostre projecte.

En aquest context, hi ha una gran quantitat de proveïdors, que, fent ús dels avantatges que representa el factor d'escala, ofereixen solucions de lloguer d'infraestructura, dedicada i compartida.

Entre les solucions de lloguer d'infraestructura podem destacar dos models, el *hosting* i el *housing*.

El *hosting* ens permet llogar una infraestructura física sobre la qual implementar el servidor i una plataforma on desplegar la nostra aplicació, és a dir, un sistema operatiu i *frameworks* que permetin la seva execució i la connectivitat necessària perquè el nostre sistema sigui accessible per a usuaris i administradors.

Els recursos assignats al sistema són compartits i no tenim control directe sobre ells, ni accés. L'administració sobre els recursos no és al nostre abast.

És possible, quan es preveu que el servidor tindrà alta càrrega i quan volem tenir el control total sobre la plataforma de desplegament, que en lloc de compartir els recursos del servidor calgui llogar un servidor dedicat. El *housing* ens permet llogar una solució completa, en què l'administració total del sistema és a càrrec nostre i l'assignació de recursos de processament, memòria i emmagatzematge són de la nostra responsabilitat.

També és útil quan, per les característiques del nostre sistema, no hi hagi una plataforma de desplegament que s'adapti amb suficiència als requisits, físics i operatius, del nostre desenvolupament. Les solucions de virtualització, com veurem més endavant, fan possible disposar d'una màquina virtual amb els recursos necessaris i de manera exclusiva per al nostre propòsit. Aquesta màquina virtual funciona amb caràcter general com una màquina física real. Ens permetrà des d'instal·lar qualsevol sistema operatiu compatible amb la plataforma de virtualització, fins a reiniciar-la sense afectar altres usuaris.

Totes les altres consideracions, energia, espai, climatització i seguretat del centre de dades, són responsabilitat del proveïdor de *housing*.

La decisió final sobre la millor opció dependrà de les característiques del sistema. El *hosting* serà la millor opció, sempre que la plataforma s'adapti als requisits del sistema. En cas que el sistema requereixi més control de la plataforma i assignació dels recursos, el *housing* es presenta com l'opció més convenient.

1.4.3. Els nous models: virtualització/IaaS/PaaS

El problema d'optimització de la infraestructura es pot abordar de diverses maneres. Per això es fan ús d'opcions que exploten la gran capacitat de càlcul dels nous processadors i la velocitat de les xarxes que permeten un accés cada vegada més ràpid a les dades. No obstant això, no es tracta d'una tasca senzilla,

més aviat es tracta d'un procés continu en què cada canvi pot afectar els altres components del sistema.

La virtualització és un mètode a partir del qual s'elimina la dependència de l'aplicació respecte als recursos físics necessaris perquè s'executi, afegint-hi una capa de programari de mediació entre aplicació i maquinari, anomenat *hipervisor*. Abstraient els elements de la màquina física, incloent-hi el processador, la memòria, l'emmagatzematge i la connectivitat, arribem a una màquina virtual (VM, Virtual Machine) suportada per un programari de virtualització.

Un sol servidor físic pot suportar diverses VM, fins a un màxim de tres-centes vint, segons la solució utilitzada, cadascuna de les quals pot funcionar sobre un sistema operatiu diferent i suportar una o més aplicacions, l'hipervisor administra les VM i els assigna recursos físics segons la demanda, i és capaç de moure les VM d'una màquina física a una altra per assegurar la disponibilitat del servei.

Un gran avantatge de la virtualització està en el fet que es fa un ús més eficient dels recursos, ja que en condicions normals hi ha períodes de temps en què el servidor està ociós i que pot ser aprofitat. D'aquesta manera, el conjunt de VM treu el màxim partit del servidor físic.

També podem esmentar les següents:

- Reducció de costos d'energia: un menor nombre de servidors físics requereixen menys energia i condicionament, cosa que representa un estalvi de costos.
- Desplegament ràpid: amb les eines actuals, la creació d'una màquina virtual, o l'ampliació d'una d'existent, es pot fer de manera ràpida i fàcil, sense necessitat d'apagar la màquina i sense tall de servei.
- Gran flexibilitat: la virtualització proveeix d'eines fàcils per adaptar els canvis, les modificacions i els nous desenvolupaments als nous requeriments, a més a més de la seva implantació en producció.
- Millor control: amb pocs servidors físics és molt més fàcil centrar-se en les tasques realment crítiques de la provisió.

La virtualització ha fet possible, des del punt de vista del proveïdor, oferir solucions a mida, beneficiant-se del factor d'escala i de la gran potència dels equips actuals i des del punt de vista del client, un estalvi considerable en infraestructura, així com una gran oferta.

En aquest sentit, ens podem plantejar: què lloguem? Hem vist que és possible llogar màquines físiques o virtuals per mantenir un control total sobre els recursos, o llogar plataformes sobre les quals *desplegar* la nostra aplicació. Aquests conceptes, units a la virtualització, han donat lloc a una nova idea de *servei*.

Infraestructura com a servei (IaaS): Consisteix en l'externalització d'alguns o tots els servidors de capes lògiques, i es pot tractar de la base de dades, d'espai d'emmagatzematge, de *front-end* o d'aplicació, basada en virtualització, en què es paga pel consum de recursos. Els recursos contractats poden ser espai d'emmagatzematge, capacitat de procés, espai de base de dades i capital de transferència de dades.

Aquesta solució proveeix *infraestructura*, és a dir, un suport maquinari, típicament virtual, sobre el qual no solament haurem de desplegar la nostra aplicació, si no totes les plataformes necessàries per a la seva execució. Només li traiem "ferro" al problema.

Entre els principals proveïdors d'infraestructura, podem esmentar:

- Amazon Elastic Compute Cloud (EC2) <http://aws.amazon.com/es/ec2>
- Flexiscale <http://www.flexiscale.com>
- RackSpace Cloud <http://www.rackspace.com/cloud>
- Arsys <http://www.arsys.es>

Plataforma com a servei (PaaS): comprèn no solament la infraestructura, si no una plataforma completa, l'entorn d'execució del sistema, servidors d'aplicació, servidors de base de dades i proveeix mecanismes de manteniment, supervisió i còpies de seguretat, en què només cal *desplegar* la nostra aplicació.

Sens dubte, la contractació de PaaS és un gran encert quan les solucions disponibles s'adapten a les necessitats de la nostra aplicació, però són molt menys flexibles que les IaaS i poden caldre solucions addicionals, amb el cost corresponent.

La contractació de PaaS deixa del costat del proveïdor el pagament de llicències i el manteniment de la plataforma.

Entre els principals proveïdors de PaaS esmentem:

- Google AppEngine <http://appengine.google.com>
- Windows Azure Platform <http//www.microsoft.com/windowsazure/>
- Force <http//www.force.com>

Programari com a servei (SaaS): Es tracta de solucions completes i *funcionals* que satisfan necessitats *específiques*.

Serveixin d'exemple Google Apps <http://www.google.com/apps/>, SalesForce <http://www.salesforce.com> i Office365 <http://www.microsoft.com/es–es/office365/online–software.aspx>.

Els últims anys, amb la consolidació d'Internet i l'expansió contínua de les xarxes d'alta velocitat, a més a més d'una contínua millora de la capacitat de procés amb l'aparició de nous processadors més potents, unitats d'emmagatzematge més grans, ràpides i sobretot, més barates, s'ha disparat el nombre de proveïdors de serveis en el núvol. Ja no cal disposar de grans infraestructures i costosos centres de dades per oferir solucions de serveis a bon preu. En aquesta mesura, els petits i mitjans proveïdors de serveis estan competint amb els grans proveïdors i guanyant el mercat de negocis de petita i mitjana càrrega, i si no necessitem terabytes d'emmagatzematge, ni 32 cores per a la nostra aplicació, són la solució més barata i més fàcil.

1.5. Arquitectura del servidor web

Quan parlem d'arquitectura d'un servidor ens referim a tres aspectes fonamentals: l'estructura dels components, les interrelacions entre ells i els principis que governen el seu disseny i la seva evolució al llarg del temps. En aquest sentit, cal considerar, les qüestions següents:

- Com podem estructurar l'aplicació per donar la màxima flexibilitat i fiabilitat?
- Quina és la infraestructura necessària per oferir una aplicació portable i interoperable?
- Com col·loquem els components de l'aplicació (programes, interfícies d'usuari, bases de dades, etc.) per maximitzar la disponibilitat i optimitzar el rendiment?

La clau està a en identificar les tasques que ha de complir el servidor i la manera com aquestes tasques es porten a terme. Concretament:

- El servidor conté dades.
- El servidor ha de ser capaç de treballar amb aquests i altres dades generant resultats.
- El servidor ha de ser capaç d'informar d'aquests resultats l'usuari.

A partir d'aquests conceptes, definim tres capes lògiques: capa de dades, capa de processament i interfície d'usuari.

El pròxim pas és organitzar i distribuir físicament aquestes capes per obtenir els nivells de rendiment, cost i disponibilitat desitjats.

Partint de la idea més simple, podríem ubicar aquestes tres capes dins d'una sola màquina, un servidor integrat que s'encarregui d'emmagatzemar les dades, processar-les i mostrar els resultats, o podríem separar la tasca de presentació de resultats de les dues altres o, finalment, podríem separar les tres capes en màquines diferents.

D'aquesta manera, podem classificar els servidors en els tipus següents:

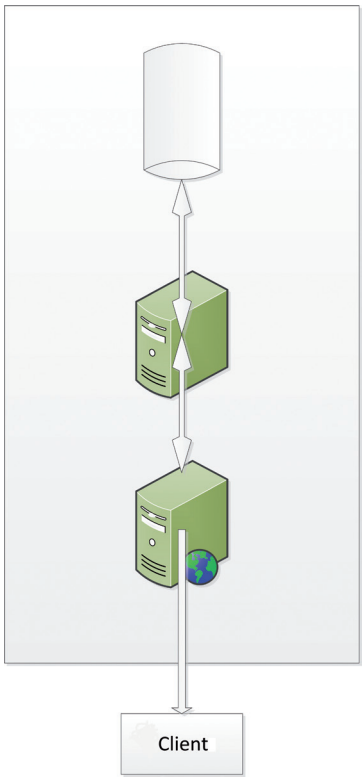
- Servidors integrats.
- Servidors en dues capes.
- Servidors en tres capes.

En un model de diverses capes, cadascuna actua com a client de les capes inferiors i com a servidor de les capes superiors; d'aquesta manera, la interfície d'usuari invoca funcionalitats de la capa intermèdia i realitza peticions. La capa intermèdia executa els processos associats al negoci, fa peticions al motor de base de dades, processa la resposta i genera resultats que torna a la interfície d'usuari, que dóna el format per a la seva presentació.

És fonamental determinar el nivell apropiat de distribució, ja que aquesta decisió impacta directament en l'elecció de la infraestructura, la implementació i el rendiment final.

1.5.1. Servidors integrats

Un servidor integrat és aquell en què les tres capes lògiques estan suportades per la mateixa capa física, és a dir, que estan en la mateixa màquina, comparteixen una mateixa plataforma (el sistema operatiu) i els recursos de què disposa la màquina, emmagatzematge, memòria i capacitat de procés.



Totes les capes lògiques sobre una mateixa infraestructura física

Un exemple d'implementació d'un servidor integrat és el servidor LAMP (Linux-Apache-MySQL-Php), que consta d'un servidor web, encarregat de rebre les peticions de pàgines, invocar l'intèrpret Php que s'encarrega de processar els paràmetres passats pel client, executar la lògica de negoci, explotar la base de dades mitjançant connexions amb el motor de BBDD MySQL, generar resultats i presentar una pàgina per al seu desplegament en el navegador client.

Són una opció viable quan la càrrega esperada sobre el sistema, o sobre algun dels elements del sistema, no és gaire alta, ja que els recursos de la màquina han d'atendre tots els processos d'aplicació, de base de dades i de presentació, i es poden generar bloquejos quan alguna de les capes lògiques consumeix tots els recursos.

La capacitat d'ampliació del servidor està determinada pels recursos de la màquina.

Entre els principals avantatges que comporta implementar totes les capes lògiques en la mateixa infraestructura física, podem esmentar les següents:

- **Menor cost inicial**: la inversió inicial és assumible per a entorns de negoci baixos o mitjans en què la càrrega total sobre el servidor, tant en continguts com en accessos d'usuari, no és excessiva. La inversió necessària per dotar d'alta disponibilitat també es redueix, com veurem més endavant.
- **Menor cost de manteniment**: s'eviten els problemes habituals de connectivitat física entre capes (caigudes d'enllaços entre el *front-end* i l'aplicació, saturació d'enllaços entre l'aplicació i la base de dades, etc.). Tot això redueix els costos de manteniment del sistema.
- **Facilitat de control i administració**: el control se simplifica, i es pot fer des d'un únic tauler de control, cosa que facilita l'administració del sistema, des de l'alta de perfils d'usuari fins a la configuració del servidor web.
- **Seguretat**: és més fàcil implementar polítiques de seguretat sobre una sola màquina que sobre una infraestructura composta de diversos elements físics i interfícies.

Fins aquí tot bé. La solució és barata d'implementar i mantenir, és segura i manejable, però hem de tenir en compte altres aspectes:

- **Cost de creixement**: la capacitat de creixement del nostre servidor es pot veure limitada per les característiques físiques i tecnològiques de la màquina, fins i tot pot no ser viable, la qual cosa ens obliga a adoptar altres solucions. Això fa que el cost d'ampliació sigui alt. Aquest problema es pot evitar fent una bona planificació inicial, i una anàlisi i una estimació de creixement del negoci periòdics.
- **Flexibilitat**: entenem per flexibilitat la capacitat del sistema d'adaptar-se als canvis que es puguin produir en el seu entorn i, en el cas particular dels servidors integrats, la flexibilitat és mínima, ja que el sistema funciona com un sol bloc.
- **Rendiment**: tal com comentem, el servidor integrat és viable per a casos de negoci de baixa càrrega. En cas que es produeixi una pujada en l'activitat sobre qualsevol de les capes lògiques, el servidor és susceptible de saturar-se, en no disposar de recursos suficients, i això afecta el sistema completament.

- **Disponibilitat**: per assegurar la disponibilitat, hi ha d'haver una màquina, generalment idèntica, de tal manera que, davant un eventual problema, el servei sigui prestat per la màquina bessona.

Alguns d'aquests aspectes seran més crítics que altres segons de les característiques del negoci, però, sens dubte, tots són importants a l'hora de dissenyar un sistema robust, òptim i tolerant a les fallades, tots influeixen directament sobre la qualitat del servei. En la taula següent tenim un resum del que hem exposat anteriorment, per tal d'establir un criteri de comparació amb les arquitectures que veurem més endavant.

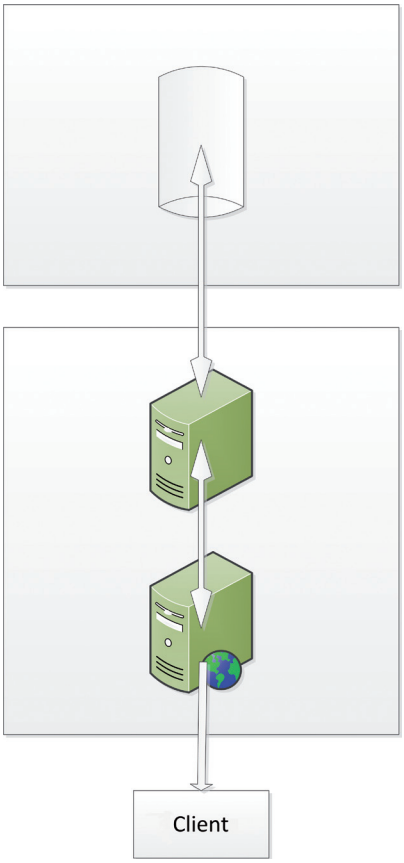
Servidor integrat	
Cost inicial	Baix
Cost de manteniment	Baix
Cost de creixement	Alt
Independència d'usuaris	Dolent
Flexibilitat	Dolent
Rendiment	Saturació del servidor
Disponibilitat	Dolent
Control i administració	Molt bo
Seguretat	Molt bo
Aplicacions típiques	Aplicacions de baixa càrreg

1.5.2. Servidor en dues capes

A l'hora de dissenyar l'arquitectura del sistema és vital identificar els punts crítics, aquells que suportaran la càrrega de processos i transaccions més elevada, i que consumiran la major part dels recursos.

Imaginem per un moment que el servidor LAMP descrit anteriorment servirà com a implementació d'un sistema la característica principal del qual és l'alt nombre de transaccions realitzades sobre la base de dades i que cadascuna d'aquestes transaccions consumeix una quantitat considerable de memòria de la màquina. En aquest cas, és convenient allotjar la base de dades en una infraestructura física a part, amb independència de recursos i aïllada, la qual cosa ens permetrà realitzar una supervisió específica per detectar sobrecàrregues i colls d'ampolla, i evitar el bloqueig o degradació dels altres components del sistema.

Per tant, tindrem un servidor en dues capes físiques: una que suporta el front-end i la lògica de negoci i una altra que suporta la base de dades:



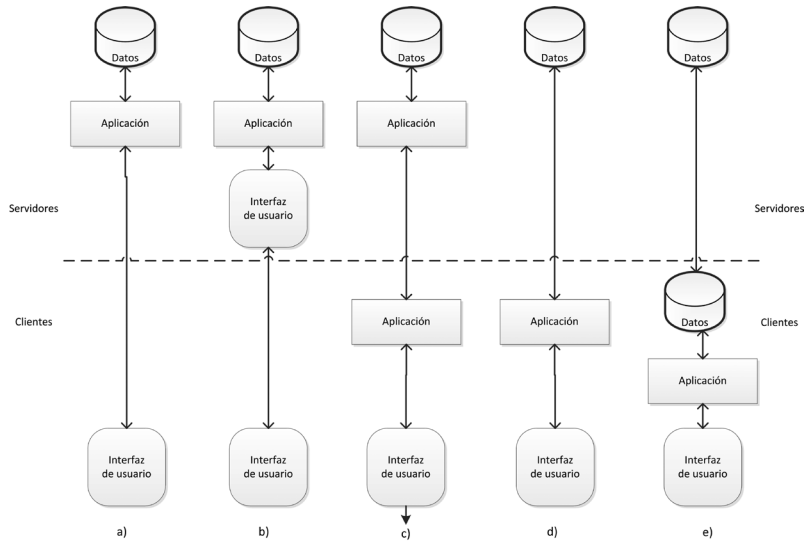
La capa de dades està separada en una infraestructura física diferent

En general, el servidor de dues capes representa una solució equilibrada en termes de cost i rendiment, i es pot implementar en negocis amb càrrega mitjana, concretament:

- **Cost inicial**: la inversió inicial és assumible per a entorns de negoci mitjans o alts amb un nombre considerable d'usuaris i complexitat de processos.
- **Cost de manteniment**: l'assegurament de la connectivitat entre la màquina de front-end i la lògica de negoci i la de base de dades afegeix un cost extra de manteniment, que és superior al del servidor integrat.

- **Facilitat de control i administració**: les tasques de control i administració es divideixen entre l'administrador del sistema i l'administrador de la base de dades.
- **Cost de creixement**: quan el creixement és vegetatiu, és a dir, que està relacionat directament amb el creixement del negoci i amb la quantitat de dades que ha de manejar, el cost de creixement és relativament baix respecte del cost inicial, si el creixement degut a noves necessitats, nous desenvolupaments o un creixement del nombre d'usuaris en el sistema el cost de creixement és similar al del servidor integrat.
- **Rendiment**: la separació de processos ens permet assignar els recursos a les tasques més crítiques, la qual cosa es tradueix en un rendiment més alt.

La separació de la base de dades en una màquina a part és una solució vàlida per als casos en què els processos de connexió i transaccions requereixen més recursos, però hi ha altres configuracions segons les característiques del sistema i l'activitat de les diferents capes lògiques i del consum:



Configuracions per a servidors en tres capes: a) Presentació remota, b) Presentació distribuïda, c) Aplicació distribuïda, d) Dades remotes, e) Dades distribuïdes

Les arquitectures a) i b) s'utilitzen en aplicacions amb més complexitat en la capa de presentació: La tercera representa l'arquitectura amb processos d'aplicació distribuïts en la qual els programes de lògica de negoci se separen entre les

màquines client i servidor i es comuniquen cadascuna amb l'altra a través de crides de procediment remot (RPC). La quarta, d), representa l'arquitectura de dades remotes en què les dades remotes estan normalment emmagatzemades en un «servidor SQL», a les quals s'accedeix mitjançant sentències SQL *ad-hoc* enviades a través de la xarxa. La e) representa el cas en què les dades existeixen tant en les màquines client com servidor (arquitectura de dades distribuïdes).

Com veiem en la taula adjunta, el servidor en dues capes compensa en gran manera l'augment de costos d'infraestructura amb una pujada important del rendiment i estabilitat del sistema:

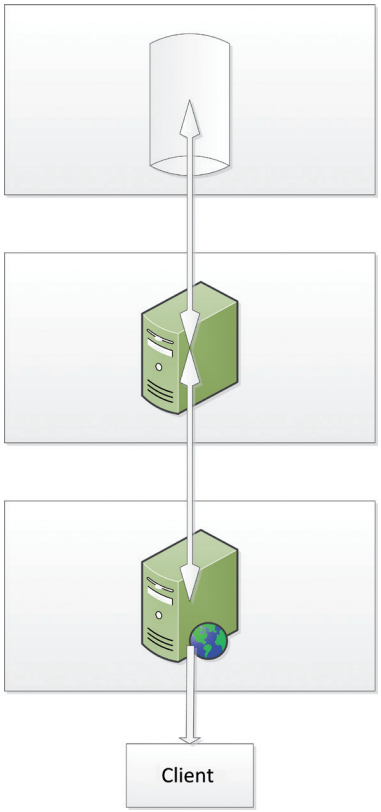
Servidors en dues capes	
Cost inicial	Mitjà
Cost de manteniment	Mitjà
Cost de crecimiento	Mitjà
Independència d'usuaris	Bo
Flexibilitat	Bo
Rendiment	Saturació a la xarxa
Disponibilitat	Bo
Control i administració	Bo
Seguretat	Bo
Aplicacions típiques	Aplicacions de càrrega mitjana

1.5.3. Servidor en tres capes

En un servidor en tres capes, les capes lògiques, de presentació, lògica de negoci i de dades, estan separades en infraestructures físiques diferents, cadascuna en una màquina real o virtual, amb els seus propis recursos i interaccionant entre si per oferir el servei.

És important que la interacció entre cadascuna de les capes es realitzi mitjançant un model client/servidor, en què unes capes ofereixen serveis a altres.

Un servidor J2EE és un exemple habitual d'un servidor en tres capes. Hi tenim un servidor de presentació com a contenidor de servlets, JSP, JavaBeans o una barreja d'aquests, basat en Tomcat o JBoss, conegut com a front-end, un servidor d'aplicació que s'executa sobre una plataforma WebSphere o Weblogic i que accedeix a la capa de dades, coneguda com a back-end, a través de solucions JAVA com JDBC o Hibernate.

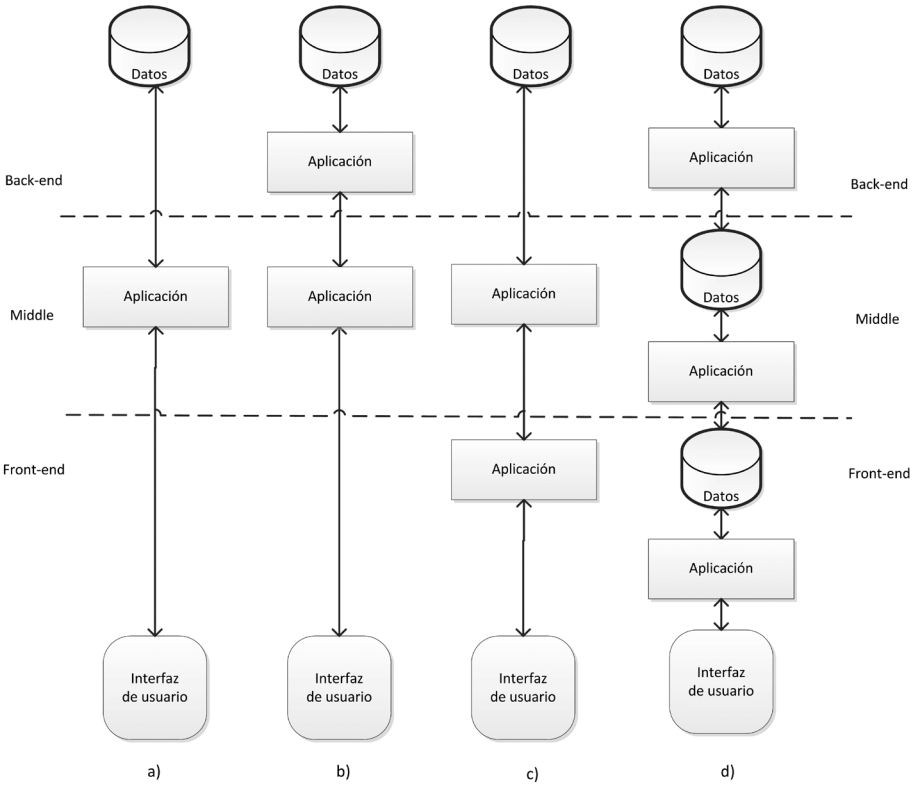


Totes les capes lògiques estan en infraestructures físiques diferents

Una arquitectura en tres capes proveeix, sobretot, de flexibilitat. Diverses configuracions són possibles, tant a nivell de servei com a nivell de seguretat i disponibilitat. És el model ideal per a sistemes amb alta càrrega i un gran nombre d'usuaris per la seva gran escalabilitat. A més a més d'aquestes, presenta les característiques següents:

- **Cost inicial més elevat**: la inversió inicial és molt més gran que en els dos casos anteriors, encara que les actuals solucions de virtualització ajuden a abaratir costos.
- **Cost de creixement**: a causa de l'alta flexibilitat del sistema el cost de creixement és molt menor que en el cas d'un servidor integrat. El gran nombre de configuracions possible ens permet dissenyar el sistema de tal manera, que el cost d'ampliació de sistema es redueixi.

- **Rendiment**: l'avantatge més destacable respecte dels dos models anteriors radica en el rendiment. Com que les capes lògiques estan separades, podem assignar de manera eficient els recursos, sempre segons la càrrega sobre el sistema i els processos crítics. També ens permet implementar solucions per evitar colls d'ampolla i assegurar la disponibilitat.
- **Disponibilitat**: molt alta. La gran flexibilitat ens permet adoptar solucions de balanceig de càrrega, alta disponibilitat, continuïtat i seguretat de dades, solucions de *backup* més eficients, etc. Tot això fa que la configuració del sistema en tres capes sigui la solució implementada en les grans aplicacions corporatives, amb un nombre alt d'usuaris i una gran necessitat de capacitat de processament.



Configuracions per a servidors en tres capes: a) Configuració estàndard. b) i c) Aplicació distribuïda. d) Aplicació i dades distribuïdes

Les configuracions de la figura mostren exemples d'arquitectures en tres capes físiques. L'arquitectura per la qual ens decidim dependrà en gran manera del tipus d'aplicació i de les característiques d'ús del sistema. Les configuracions a) i b) són usades en qualsevol aplicació a gran escala. Per exemple, botigues en línia <http://www.amazon.com> o amb un gran volum d'informació <http://www.wikipedia.com>, <http://www.ebay.com>. La configuració c) s'utilitzaria en casos en què cal una aplicació distribuïda entre servidor i client (per exemple, Spotify), i la configuració d) s'utilitzarà en sistemes molt específics (per exemple, sistema de venda i facturació d'El Corte Inglés).

Servidors en tres capes	
Cost inicial	Alt
Cost de manteniment	Alt
Cost de creixement	Baix
Independència d'usuaris	Molt bo
Flexibilitat	Molt bo
Rendiment	Molt bo (diverses posibilitats)
Disponibilitat	Molt bo
Control i administració	Dolent
Seguretat	Dolent
Aplicacions típiques	Aplicacions d'alta càrrega

1.5.4. Mètodes de connexió amb les bases de dades

La solució més senzilla per a l'accés a la base de dades des de l'aplicació consisteix a crear una connexió en iniciar l'aplicació i utilitzar aquesta connexió cada vegada que es requereixi fer una transacció. El consum de recursos és baix i la connexió es pot mantenir oberta tot el temps que calgui, típicament, i la connexió es tanca quan l'aplicació acaba.

Això presenta un problema, l'accés a la base de dades és seqüencial, no podem fer multitasca ni múltiples accessos. En servidors web aquest problema es torna crític, ja que tenim múltiples usuaris sol·licitant informació de la base de dades i no és lògic que un usuari hagi d'esperar que la connexió quedi lliure, això crearia colls d'ampolla en l'accés a base de dades que farien inviable la solució.

Una altra possibilitat és crear diverses connexions segons que calguin. Un usuari sol·licita accés a la base de dades i es crea una connexió; totes les transaccions de l'usuari utilitzen la connexió, i després, quan ja no cal, es tanca la connexió. Aquesta solució ens permet fer consultes en paral·lel i és òptima en els casos en què el nombre d'usuaris que accedeixen simultàniament a la base de dades no és alt. Hem de considerar que la creació i el tancament de connexions amb la base de dades és una operació costosa en termes de recursos màquina.

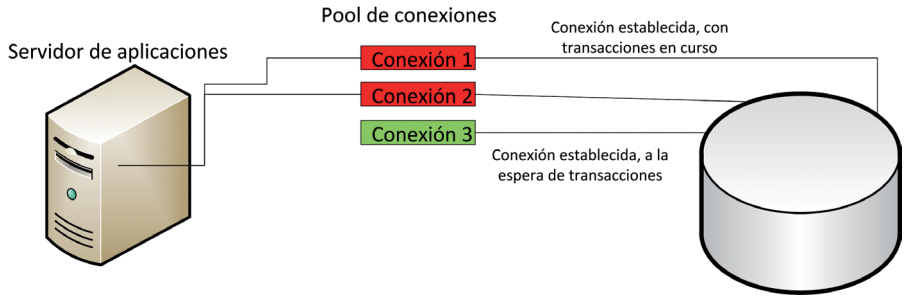
El pas següent en els mètodes d'accés a la base de dades és el *pool* o conjunt de connexions. Quan s'inicia l'aplicació es creen connexions sota demanda, i quan les connexions ja no calen, no es tanquen, sinó que romanen disponibles fins a la següent petició, quan un altre accés és requerit. L'aplicació s'ocupa de posar a disposició una connexió vàlida del conjunt de connexions emmagatzemades. Si totes les connexions estan ocupades i no queden connexions lliures, llavors es crea una nova connexió i s'afegeix al *pool*. Una vegada finalitzades les transaccions sobre la connexió, aquesta no es tanca, sinó que és emmagatzemada dins del *pool* fins que una nova petició requereixi una connexió lliure.

El *pool* de connexions, a més de representar una solució eficient en l'ús de recursos de la màquina, també és una solució òptima en rendiment, ja que el temps que roman l'usuari a l'espera d'una connexió lliure es redueix considerablement.

Els paràmetres associats al *pool*, com el nombre mínim i màxim de connexions i el nombre de connexions ocupades, són fàcilment configurables per garantir un funcionament correcte i adequat als requeriments d'ús del sistema.

És habitual l'ús del *pooling* de connexions en aplicacions corporatives basades en web i manejades per un servidor d'aplicacions, però el seu ús no es limita únicament a aquests, totes les aplicacions que requereixin un ús freqüent d'accessos a base de dades poden utilitzar el *pooling* per reduir temps d'espera i fer un ús eficient dels recursos, encara que tant el motor de base de dades com la mateixa aplicació han d'implementar recursos de programari necessaris per al maneig de *pools*. Actualment s'ha reduït molt la complexitat de les aplicacions gràcies a les solucions de programari que han creat tercers i grups de desenvolupadors que fan possible el maneig de *pools*.

El *pooling* de connexions està suportat pels principals motors de base de dades del mercat, DB2, Microsoft SQL Server, Oracle, MySQL i PostgreSQL.



Les connexions sense transaccions en curs romanen actives i a l'espera

1.5.5. Escalabilitat i alta disponibilitat

Totes les tasques anteriorment realitzades, d'anàlisi de càrrega, de definició dels components del sistema i de l'arquitectura ens permetran complir els requeriments actuals de servei i de qualitat de servei a curt termini. No obstant això, és imprescindible fer una estimació de l'evolució a mitjà i llarg termini, ja sigui en nombre de clients, en capacitat de processos o en creixement vegetatiu de la base de dades, i facilitar els mecanismes necessaris per a una possible ampliació, per tal de mantenir la qualitat de servei en el nou entorn.

Definim *escalabilitat* com la capacitat del sistema d'evolucionar per adaptar-se a nous requeriments, ja sigui mitjançant la implantació de nous servidors d'aplicacions, de bases de dades o de *front-ends*, com l'ampliació de les màquines ja implantades.

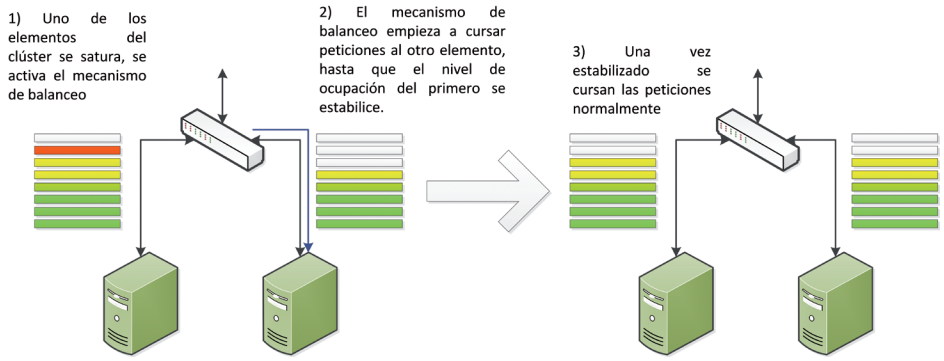
Prenent com a referència el servidor en tres capes, identifiquem els punts crítics del sistema susceptibles de saturació que poden ser el *front-end* (capa de presentació), el servidor d'aplicacions (capa de lògica de negoci) i la base de dades (capa de dades) i presentem com a solucions d'escalabilitat el balanç de càrrega, l'ús de clústers i el *cloud computing*.

1.5.5.1. Balanceig de càrrega

Un balancejador de càrrega és un element configurable, que pot ser tant de maquinari com de programari, encarregat de repartir les peticions de servei entre els recursos de maquinari disponibles tal que el nivell d'ocupació d'aquests sigui tan equilibrat com es pugui, per tal d'evitar la saturació i la conseqüent degradació del servei.

L'algoritme implementat per al balanceig de càrrega pot ser des del més senzill, per exemple, de tipus Round-Robin, o algoritmes més complexos segons les característiques de les peticions i dels elements de processament disponibles.

El balanç de càrrega es presenta com la solució més comuna en entorns d'alta càrrega que fan un ús intensiu dels recursos, habitualment els *front-ends*.



Balancejador de càrrega

1.5.5.2. Clústers

Un clúster és un conjunt d'elements del sistema l'objectiu principal del qual és assegurar la disponibilitat del sistema.

Els elements que conformen el clúster poden ser idèntics, tant en rendiment com en SO. En aquest cas s'anomenen homogeni; poden tenir un rendiment diferent i el mateix SO, i s'anomenen semihomogenis, o ser diferents tant en rendiment com en SO, i s'anomenen heterogenis.

Hi ha dues configuracions possibles segons l'objectiu crític del sistema: una ens permet assegurar que el servei s'ofereix de manera òptima en rendiment i l'altra ens assegura alta disponibilitat.

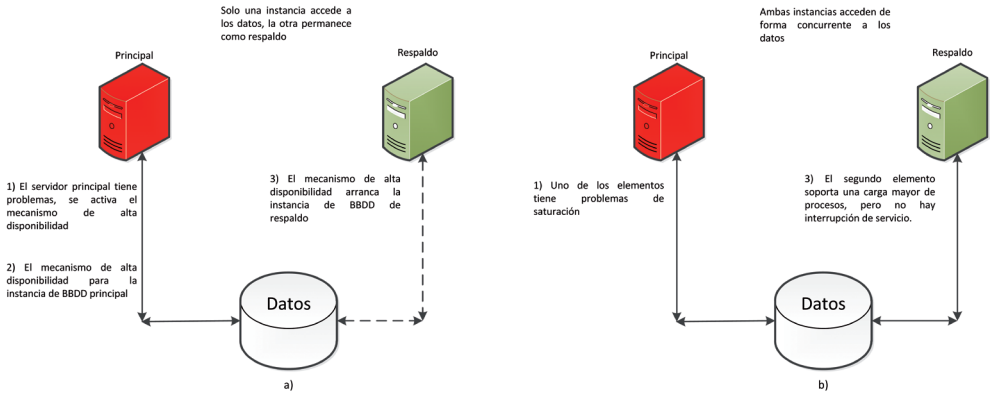
Clústers actiu-actiu: en un clúster actiu-actiu, tenim elements, típicament *front-ends*, treballant en paral·lel. Els dos servidors tenen accés als altres recursos del sistema i poden interactuar amb la capa de lògica de negoci de manera simultània.

La configuració de clúster actiu-actiu requereix que l'aplicació estigui preparada per a aquesta concurrència, per la qual cosa, des del punt de vista de desenvolupament de l'aplicació, pot implicar costos més alts.

Els elements que conformen el clúster poden residir a la mateixa ubicació geogràfica, fins i tot dins d'una mateixa cabina o estar distribuïts per assegurar redundància geogràfica, molt útil per evitar possibles problemes de connectivitat i energia en els CPD.

Els recursos individuals de cadascun dels elements es presenten com un sol recurs global, disponible per servir les peticions del balancejador, en cas que estiguin separats geogràficament, o del programari controlador de la cabina, en cas que estiguin en la mateixa ubicació.

Clústers actiu-passiu: aquesta configuració ens permet disposar d'un suport que s'activarà en cas d'indisponibilitat del servidor principal, si bé els recursos del suport no estaran disponibles fins al moment en què es commuti el servei d'un servidor a l'altre. Aquesta commutació comporta un temps d'indisponibilitat curt que interessa minimitzar al màxim.



Alta disponibilitat amb clústers aplicats a base de dades: a) clúster actiu-passiu, b) clúster actiu-actiu

Encara que a priori, el model de clúster actiu-actiu permet aprofitar millor els recursos disponibles, l'aplicació ha d'estar dissenyada per a aquest efecte. Aquesta característica es coneix com a «clúster-aware», els processos han de ser concurrents i les despeses assignades per al desenvolupament de l'aplicació s'incrementen.

Actualment, hi ha diverses solucions que ens permeten dotar d'alta disponibilitat els elements del sistema, ja sigui el servidor d'aplicacions, el *front-end* o la base de dades, entre els més importants podem destacar:

HP-ServiceGuard: es tracta d'una solució per a la implementació de models actiu-passiu. Aquesta eina s'ha de configurar per executar ordres de parada i arrencada (típicament *scripts*) dels servidors principal i suport, de tal manera que quan hi hagi una interrupció en el servei, per indisponibilitat del servidor principal, ServiceGuard s'ocupa de parar totes les instàncies que s'estiguin executant i posa en marxa les instàncies del servidor de suport. D'aquesta manera es produeix una commutació del servei i encara que hi hagi un tall d'aquest, sol estar dins dels nivells d'acord contractats pel client.

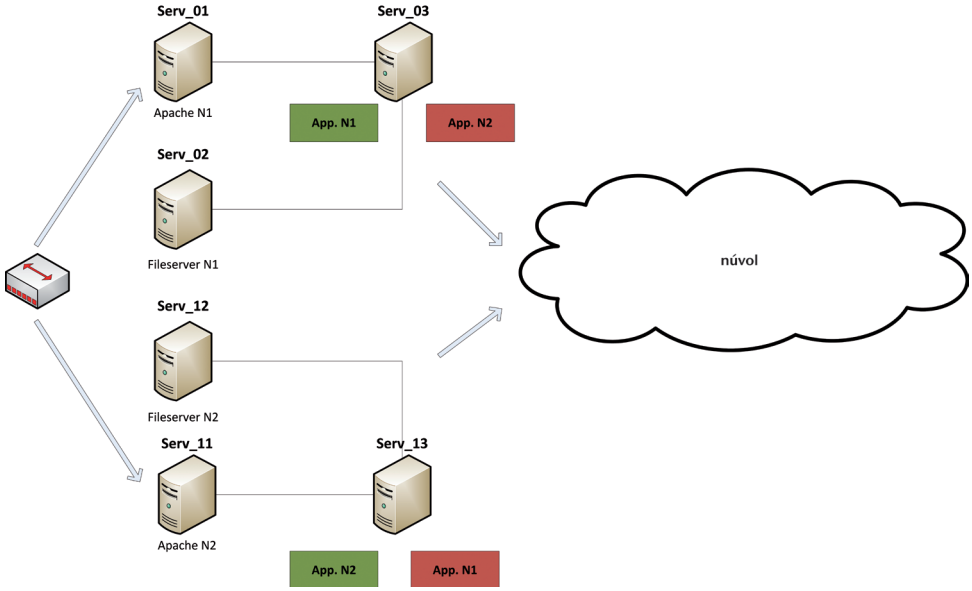
Oracle-RAC: es tracta d'una solució per a la implementació de models actiu-actiu en bases de dades. Típicament el servidor que allotja la base de dades està connectat amb dispositius d'emmagatzematge amb redundància (per exemple, SYMMETRIX) per garantir-ne la integritat; doncs bé, Oracle-RAC permet que dues instàncies del motor de base de dades accedeixin simultàniament als dispositius d'emmagatzematge, de tal manera que el contingut de tots dos sigui sempre el mateix. Quan un d'ells deixa d'estar disponible, les dues instàncies del motor apunten al que està disponible; d'aquesta manera no hi ha tall de servei.

1.5.5.3. El servidor web en el núvol

El model de *cloud computing* es caracteritza per oferir serveis, ja sigui serveis de programari, serveis de plataforma o serveis de infraestructura, que permeten als usuaris, amb un grau mínim d'experiència, disposar dels recursos necessaris per implantar el negoci.

Els serveis es caracteritzen per oferir solucions flexibles, adaptatives i escalables de manera ràpida, eficient i transparent per a l'usuari, a més d'oferir disponibilitat global gràcies a l'ús estès d'Internet.

En el cas particular dels servidors web, el model en el núvol ens permetria contractar el lloguer de la infraestructura física, un servidor amb una amplada de banda i un volum de dades concret, compost de tres màquines virtuals que implementin les tres capes lògiques, o un servidor dedicat per a la base de dades, per ser més concrets, tornem a l'exemple de l'apartat 1.4 i vegem el que el *cloud computing* ens pot oferir:



Servidors de bases de dades i solucions d'alta disponibilitat, redundància i emmagatzematge en el núvol

Aquesta solució és possible, per exemple, contractant serveis d'infraestructura per a emmagatzematge i serveis de plataforma per als servidors de bases de dades.

Sempre que els serveis s'adaptin a les necessitats del nostre sistema, podríem portar el concepte del *cloud* a la màxima implementació possible, contractant PaaS per als servidors d'aplicació, front-ends i bases de dades, SaaS per als servidors de fitxers i IaaS per a l'emmagatzematge. D'aquesta manera tot el nostre sistema estaria en el núvol, i ens preguntarem, on estaria el nostre sistema…? Mai no ho sabrem. És possible que els nostres servidors de base de dades estiguin geogràficament distribuïts i el nostre servidor d'aplicacions a l'altra banda del món. El que és important és que funciona.

Aquesta solució és viable sempre que el negoci sigui capaç d'afrontar els costos de manteniment, que en aquest cas es tradueixen en el lloguer dels serveis.

Capítol II

L'entorn de producció

Héctor Alonso Martín

Després d'estudiar en el capítol anterior les definicions bàsiques dels components que formen un entorn web, en aquest capítol passarem a desenvolupar les diferents etapes que comporten la planificació, instal·lació i posterior explotació de l'entorn. Arribats a aquest punt, és molt important tenir definit clarament el projecte al qual ens enfrontem. Per saber si tenim totes les dades bàsiques que ens faran falta al llarg de la vida del projecte, hem de poder contestar les preguntes inicials següents:

- Quin tipus de projecte farem?
- Quin és l'objectiu del projecte?
- A quants usuaris potencials s'adreça?
- Quants usuaris concurrents estimem tenir connectats a l'entorn?
- Quin tipus i quantes dades hem de tractar?
- Quina ha de ser la disponibilitat de l'entorn?

Dins de la realització d'un projecte, hi ha moltes altres preguntes bàsiques referents a costos, viabilitat, escalabilitat... Totes aquestes qüestions són tan importants o més que les anteriors, però per simplificar el problema i poder-nos centrar més en l'objectiu del capítol, partirem d'una base utòpica en què disposem del pressupost necessari i la infraestructura adequada per poder desenvolupar tot el projecte segons les necessitats que detectem i poder aportar-hi la millor solució.

2.1. Instal·lació dels components d'un servei web

Quan afrontem el repte d'engegar un servei web, podem dividir les tasques necessàries en quatre grans apartats fonamentals: disseny, planificació, installació i desplegament i gestió de la configuració. Aquestes tres etapes són molt

importants i es complementen, per la qual cosa si cometem errors en la primera, poden ser arrossegats a les següents, i haurem d'invertir temps i diners a corregir un error que podríem haver evitat dedicant un poc més d'atenció al principi. Abans de començar l'etapa de planificació, hem de contestar les preguntes plantejades inicialment en el projecte per tal de poder fer un calendari i marcar unes fites precises i assumibles. Intentarem respondre d'una manera general les preguntes anteriors i així entendre per a què ens serveixen i com ens marcarà cadascuna:

Quin tipus de projecte farem?

Una part molt important per a nosaltres a l'hora de definir que ens farà falta o com plantejar el desplegament d'un servei web és entendre perfectament el que hem de fer i l'objectiu final del que pretenem muntar. Quan se'ns plantegi la problemàtica inicial del projecte, l'hem d'entendre i n'hem de desprendre els components que ens faran falta per posar-la en marxa i aconseguir els objectius marcats. Així, per a un projecte senzill, com una pàgina de presentació d'una empresa en la qual només tindrem informació d'aquesta empresa, només necessitarem un servidor web, per a un de més complex, com el portal d'un banc, necessitarem balancejadors de càrrega, servidors web, servidors d'aplicacions, base de dades... Com es pot deduir, una comprensió errònia de la complexitat del projecte des de l'inici ens pot portar a oblidar-nos de parts fonamentals per al desenvolupament correcte i la posada en marxa del servei web sol·licitat, o a no tenir-les en compte.

Quin és l'objectiu del projecte?

Encara que la nostra tasca consisteixi en la planificació i posada en marxa d'un servei web i això pugui tenir certa independència del contingut de l'aplicaició o de les aplicacions que el compondran, el fet de conèixer l'objectiu i per a què s'utilitzarà el que hem de «servir» ens ajudarà a fer-nos una idea general de les fases o parts del desplegament i la instal·lació del servei en què haurem de fer més èmfasi i d'altres en què no caldrà esmerçar-hi tant de temps. Al seu torn, ens facilitarà localitzar els punts del nostre sistema que estaran més carregats de treball i que seran més susceptibles a possibles fallades. En definitiva, saber per a què servirà el que estem fent ajudarà molt que la nostra planificació inicial sigui correcta i òptima amb vista a la instal·lació i l'explotació del servei web.

A quants usuaris potencials s'adreça?

Una part fonamental en la definició i el disseny d'un servei web és conèixer la capacitat i escalabilitat que aquest ha de tenir. Així doncs, segons el tipus de projecte, ens pot importar més o menys els usuaris potencials als quals s'adreça. Per exemple, una pàgina en què els usuaris han de ser anònims no té la mateixa càrrega de dades i nivells de seguretat que una en què els usuaris s'hauran de registrar per poder accedir-hi. En el moment en què l'aplicació o les aplicacions que despleguem necessitin obtenir i desar dades dels usuaris que s'hi connectin, hem de fer una bona planificació de la capacitat del servei web i, per això, necessitarem conèixer una dada aproximada dels seus usuaris potencials. En aquests casos, la capacitat es referirà més a la quantitat d'informació que hauré de desar que al nombre de components o a la potència d'aquests que necessitaré.

Quants usuaris concurrents estimem tenir connectats a l'entorn?

El bon funcionament d'un servei web es pot mesurar per molts factors que analitzarem més endavant, però un de molt important és la qualitat de l'experiència de l'usuari a l'hora d'utilitzar-lo. Per aconseguir una bona resposta del nostre servei web amb vista als usuaris, és fonamental que estigui ben dimensionat. Conèixer el nombre estimat d'usuaris concurrents que tindrà el nostre sistema és un punt fonamental per poder definir quants components ens faran falta en cadascun dels blocs que comprèn el nostre servei i la potència que hauran de tenir aquests. En molts projectes és difícil conèixer aquesta dada i, per tant, haurem de partir d'un nombre estimatiu que ens obligarà a definir un pla sòlid d'escalabilitat del servei web. No hem d'oblidar mai que un projecte ve molt marcat pels costos i que aquest serà un cavall de batalla important pel que fa als temes de capacitat i escalabilitat; per tant, tenir una bona planificació inicial d'aquests conceptes ens estalviarà problemes al llarg de la vida de la implantació i explotació.

Quin tipus i quantes dades hem de tractar?

En la resposta a quants usuaris potencials tindrà el nostre projecte ja vam introduir la importància de conèixer quanta informació i de quin tipus necessitarem emmagatzemar per al funcionament del servei. Les dades poden marcar molt el plantejament del nostre servei, fins al punt que hi podrem afegir un component nou, la base de dades. En projectes senzills en què no hem de desar

dades o la quantitat d'aquestes és molt petita, pot ser que la mateixa aplicació s'encarregui de mantenir les dades mitjançant fitxers i, d'aquesta manera, es pugui prescindir de mecanismes més complexos i costosos. Per contra, en projectes en els quals el volum d'informació que manegem sigui gran o molt dinàmic és molt recomanable, o fins i tot imprescindible, disposar de bases de dades més completes i complexes.

D'altra banda, també és molt important conèixer la criticitat i/o confidencialitat de les dades que hem d'emmagatzemar, ja que haurem de pensar en mecanismes que en garanteixin la integritat i seguretat. Segons el tipus de dada que desem, haurem de saber si estan legislades d'alguna manera en el país on realitzem el projecte i, en conseqüència, posar els mecanismes adequats per complir aquesta legislació. Això pot repercutir d'una manera molt important en el nostre disseny inicial del projecte i, per tant, ho hem de tenir en compte en iniciar-lo.

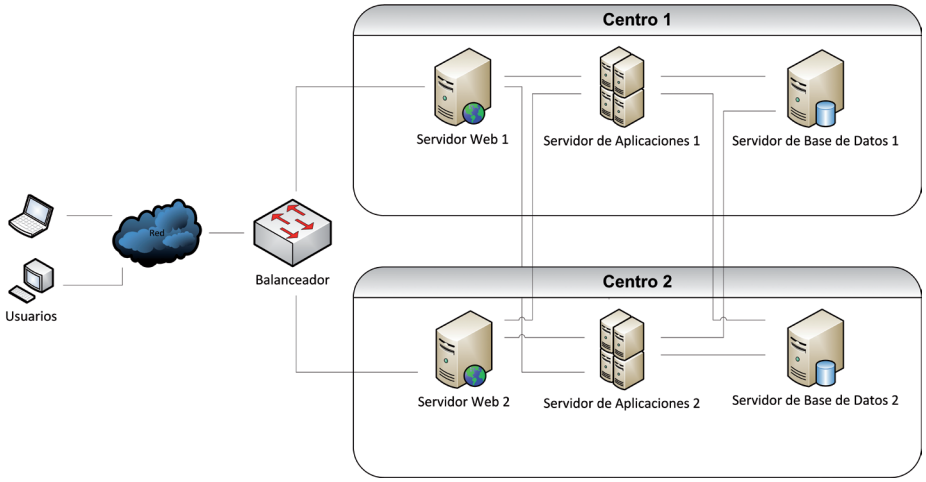
Quina ha de ser la disponibilitat de l'entorn?

Probablement amb les dues preguntes inicials ja ens hàgim fet una idea de l'ús del servei web i, per tant, de quina ha de ser la seva disponibilitat. De tota manera, és important deixar clar des d'un començament uns punts importants que marcaran profundament el disseny del servei, com ara la tolerància a fallades que ha de suportar el sistema, quant de temps podem estar sense donar servei (si podem), fins a quin punt ens hem d'anticipar a les fallades i a quin tipus d'aquestes fallades ens podríem enfrontar (caigudes de xarxa elèctrica, desastres naturals…). El nivell de servei que s'usa per mesurar la qualitat d'un servei web té com a punt fonamental els temps d'indisponibilitat del servei, per la qual cosa és obligatori tenir ben definit el que s'espera del nostre servei i el que hem de poder oferir amb les eines de què disposem.

2.1.1. Disseny del servei web

Una vegada contestades les preguntes anteriors, tindrem una idea força precisa sobre les necessitats del nostre servei web i, per tant, podem començar a fer-ne el disseny. Per començar amb el disseny, hem de tenir clarament definits els diferents components que usarem en el nostre servei per satisfer les necessitats del projecte. En el primer capítol ja hem definit els diferents elements que ens po-

drem trobar en un servei web i en aquesta part veurem com els adaptem a les nostres necessitats i com interactuen els uns amb els altres. A continuació, anirem desplegant el disseny, analitzant els diferents components segons la seva importància i funció. Com que molts dels elements estan fortament relacionats entre si, la divisió no sempre serà senzilla i veurem que la decisió sobre quin component triar moltes vegades no marcarà només el component en si, sinó també els altres components que necessitarem.



En la imatge l'esquema d'un sistema de servei web amb tots els seus components i redundància total entre ells i geogràfica

2.1.1.1. Infraestructura i arquitectura del servei

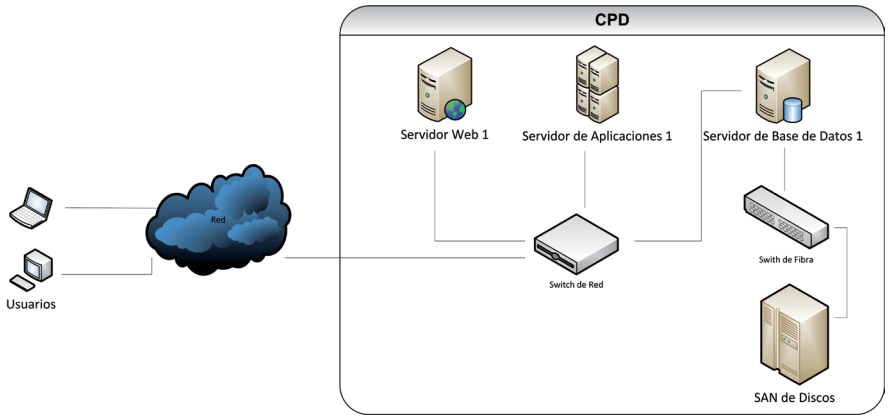
La base de tot projecte web, on es recolzaran la resta de components d'aquest, és la infraestructura i l'arquitectura de maquinari que triem. Seran fonamentals i determinants per al nostre projecte les decisions que prenguem en l'elecció dels components necessaris per a la topologia de maquinari en què es basi la nostra instal·lació. Per definir el disseny adequat hi intervindran molts factors, alguns de determinants i altres d'opcionals o variables. Intentarem descriure les opcions fonamentals que se'ns plantejaran en cada projecte web que implementem i que ens puguin portar a decantar-nos per una o una altra segons les necessitats d'aquest:

- Infraestructura pròpia: aquesta opció acostuma a ser la més «conservadora» per a les empreses, ja que tots els components de maquinari són propietat d'aquestes i, per tant, tenen un control total sobre ells. El factor fonamental que ens pot portar a decantar-nos per aquesta opció és que qui ens encarregui el projecte ja disposi de la infraestructura on aquest es desplegarà. En aquest cas, partirem de la base que els components de maquinari, i fins i tot l'arquitectura de sistema operatiu, ens vindran marcats des del principi i influiran la resta de components que hàgim d'elegir per a l'acompliment del nostre disseny. Un altre factor que influirà en aquesta elecció poden ser les polítiques de protecció de dades o de seguretat de l'empresa per a la qual fem el projecte. En cas de decantar-nos per aquesta opció, hem de tenir molt en compte els costos i tots els components que ens faran falta i dels quals no disposem. Aquests últims factors també seran determinants, ja que normalment aquesta opció requereix una inversió inicial molt més gran que les altres i que, segons les nostres necessitats presents i de futur, a la llarga, pot ser rendible (possible reutilització de components per a nous projectes, previsions de creixement…) o no. Al seu torn, dins d'aquesta opció tenim dues alternatives:
  - CPD propi: a més dels components de maquinari, també hem de tenir present que aquests han d'estar ubicats en un lloc concret que compleixi les seves necessitats bàsiques (energia, refrigeració, fonts de corrent redundants, connexions de xarxa, SAI…) Si per al nostre projecte ja comptem amb aquest espai, el més probable serà utilitzar-lo per ubicar la infraestructura que comprem. En cas d'haver de comprar o llogar aquest espai, hem de tenir en compte les tasques de manteniment i instal·lacions que requerirà i que no solament influiran en els costos inicials del projecte, sinó també al llarg de la vida d'aquest.
  - CPD aliè o *housing*: en aquest cas, una empresa externa s'encarrega de les instal·lacions i de complir les condicions necessàries per allotjar els components que necessitem. Implica una despesa fixa per al projecte, però ens estalvia un desemborsament inicial important si no disposem de l'allotjament esmentat. Si ens decantem per aquesta alternativa, hem de tenir presents les condicions i els avantatges que ens ofereixin els diferents arrendadors, veure si compleixen els nostres requisits i si tenim capacitat de creixement, en cas que calgui en un futur.

- Infraestructura llogada (*hosting*): en aquest cas, els components que necessitem, i l'espai on aquests estaran ubicats, ens seran facilitats per una empresa aliena. Una de les coses que ens pot fer inclinar per aquesta opció és el factor econòmic; ens allibera de la forta inversió inicial, però no hem d'oblidar que suposarà una despesa fixa al llarg de la vida del projecte. Malgrat la despesa fixa que suposa, dins els diferents models que tenim al mercat, podem optar per alliberar-nos de les tasques de manteniment, no solament de maquinari, sinó també a nivell de sistema dels components de la infraestructura, i per tant podrem englobar aquesta despesa dins el fix del lloguer. Si el nostre projecte té una llarga projecció de futur, amb aquesta opció ens serà molt menys costosa la renovació de la nostra infraestructura quan aquesta arribi a l'obsolescència.

Una vegada que ens hàgim decantant per una de les dues alternatives anteriors, podrem iniciar la tasca d'elecció dels components que necessitem per al nostre projecte, ja sigui per a la seva compra o dels requisits a l'hora de llogar, adequant-nos, a més a més, a les limitacions que ens imposi la decisió anterior. Aquests són els components de maquinari que poden estar presents en una infraestructura web, segons de la seva complexitat:

- Balancejador de càrrega: com hem comentat anteriorment, aquest component pot ser de maquinari o de programari, i caldrà en tots els projectes que requereixin tenir més d'una instància d'arrencada del servidor web per proporcionar els mateixos serveis.
- Components de la xarxa: dins d'aquest apartat entraran els diferents dispositius necessaris per a la infraestructura de la xarxa o les xarxes que necessitem en el nostre projecte. Podem englobar dins d'aquest apartat els *switches* per a la separació de zones de xarxa (si ens interessa) i la interconnexió dels diferents components d'aquesta; tallafocs, amb els quals podrem aïllar en diferents zones els elements de la nostra infraestructura, i salvaguardar els més crítics amb un accés segur i controlat cap a aquests; cablejat per a la interconnexió dels diferents elements… Cal tenir present que aquests elements també són susceptibles d'error i que, per tant, en entorns d'alta disponibilitat, ens interessarà tenir molts d'ells redundats per tal de no generar fallades en el servei a causa d'aquests.



En la imatge, un sistema connectat a un **switch** *de xarxa i amb el seu servidor de bases de dades connectat a un sistema d'emmagatzematge SAN mitjançant un switch de fibra*

- Emmagatzematge: en cas que en el nostre projecte calgui desar una important quantitat de dades i necessitem components externs als discos dels servidors, haurem de tenir en compte les diferents opcions que hi ha en el mercat per desar i compartir les dades. Hem de tenir present si necessitem comprar una cabina de discos o si ja en disposarem en el CPD on desplegarem la nostra infraestructura. Al seu torn, aquestes cabines se solen connectar per fibra òptica als servidors, i per tant necessitarem targetes controladores per a aquesta tecnologia en aquests, i també *switches* de fibra òptica on connectar els diferents components que requereixin accés a les dades. També ens podrem inclinar per un emmagatzematge en xarxa, on el dispositiu que emmagatzema les dades estarà connectat a una xarxa (normalment dedicada) a la qual també hauran d'estar connectats els diferents components que hi requereixin accés. Aquesta última alternativa influirà d'una manera notable en els components de xarxa que necessitem, i per tant l'haurem de tenir present a l'hora de triar-los.
- Servidors: aquest és el component més complex i en el que més ventall de possibilitats tenim. La seva elecció ens condicionarà de manera molt important la resta de decisions de disseny, ja que l'arquitectura de maquinari que triem ens limitarà en el tipus de sistema operatiu que podem instal·lar-hi i, al seu torn, en la resta de components que hàgim d'albergar en els servidors esmentats. De manera contrària, hem de tenir en

compte requisits marcats amb vista a la resta de components web que ens limitin les opcions a l'hora de triar el maquinari que necessitem. Un altre factor important que influirà de manera notòria en el nombre de servidors que necessitarem serà el nivell d'aïllament o separació que volem fer sobre els diferents components del servei (per exemple, ens pot interessar, per motius de seguretat i/o rendiment, tenir el servidor web en un servidor físic i el de base de dades en un altre). També influirà notablement en el nostre disseny el nivell de disponibilitat i redundància que necessitem per al projecte, això influirà de manera considerable en el nombre de servidors físics que requerirem per garantir una alta disponibilitat arribat el cas.

- Dispositius de còpies de seguretat: això serà fonamental en cas que el nostre projecte contingui informació crítica i molt recomanable, encara que no tinguem aquest tipus de dades, per desar la configuració i altres fitxers importants per al nostre servei. Hem de conèixer si on allotjarem els nostres components ja disposa d'aquest tipus de servei i, per tant, només els hi hem d'integrar, o si, per contra, l'hem de comprar o llogar-lo. Al mercat hi ha diverses solucions per a aquesta tasca que s'adaptaran a les nostres necessitats segons de la freqüència i quantitat d'informació que volem salvar.

2.1.1.2. El servidor web: necessitats i disseny

Per definició, dins d'un servei web, aquest és l'element fonamental i el que necessàriament hem de tenir en qualsevol disseny. En el mercat, podem trobar un variat ventall de possibilitats per a l'elecció del nostre servidor web; a continuació, intentarem tractar els punts més rellevants que ens poden ajudar a triar entre l'un o l'altre segons les necessitats que tinguem:

- Arquitectura: de vegades podrem basar l'elecció de l'arquitectura de maquinari en funció dels requisits de programari, com ara el servidor web que hem d'usar, però altres vegades, el maquinari ens vindrà ja marcat i haurem d'adaptar la resta de les nostres decisions a aquest requisit. Per tant, hem de saber la compatibilitat, tant respecte del maquinari com del sistema operatiu dels servidors web als quals optem i descartar els que no compleixin aquest requeriment.

- Robustesa: una de les característiques fonamentals en què ens hem de fixar a l'hora de decidir-nos per un servidor web és la seva fiabilitat i robustesa. Quan dubtem entre diversos possibles candidats, haurem d'analitzar estudis sobre la tolerància a fallades, proves d'estrès o de capacitat que s'hagin fet en les mateixes condicions sobre els candidats que tenim. Normalment, com més conegut i usat sigui el servidor, més informació en podrem trobar.
- Capacitat: normalment la capacitat d'un servidor web està estretament lligada al maquinari on estigui instal·lat i a la xarxa a la qual estigui connectat. Per veure el servidor que ens ofereix millors resultats, podem analitzar dades com ara la quantitat de peticions per segon que és capaç d'atendre per a unes mateixes condicions fixades. Podem buscar estudis realitzats sobre unes especificacions en diferents servidors web, la qual cosa ens ajudarà a decantar-nos per la millor opció.
- Seguretat: un altre factor important que s'ha de tenir en compte és la seguretat que ens aporta el servidor triat. Segons les necessitats del nostre projecte, haurem d'analitzar acuradament les vulnerabilitats que pugui tenir el servidor triat o la freqüència amb la qual es publiquen pedaços o versions noves que cobreixin noves vulnerabilitats o defectes. És important que el servidor que triem tingui un suport actiu per part dels seus creadors.

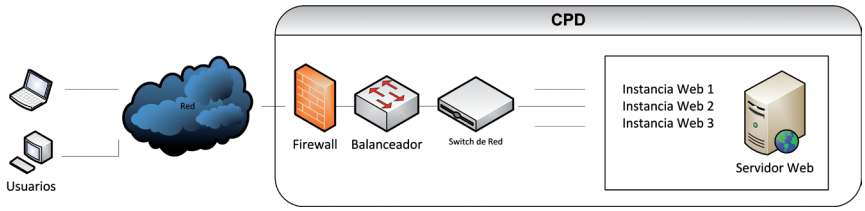
  D'altra banda, pot ser que alguna de les aplicacions que tinguem en el nostre servei web requereixi l'ús de Secure Sockets Layer (SSL) i, per tant, hem de saber si el servidor que hem triat implementa aquesta característica (actualment tots els servidors web més coneguts compleixen aquest requisit).
- Llenguatges de programació: com ja sabem, els servidors web poden executar trossos de codi de les aplicacions per servir el resultat HTML al client. Aquest procés el poden fer de manera nativa o mitjançant l'ús de llibreries externes que realitzin el processament del codi. Segons quin tipus d'aplicacions hàgim d'albergar en el nostre servidor web, hem de tenir en compte si parts d'aquestes s'executaran en la banda del servidor i, si és d'aquesta manera, conèixer el llenguatge en què estan escrites per saber si el servidor que triem tolera la seva execució. Per tant, serà un punt molt important i excloent conèixer si els servidors que contemplem admeten l'execució del codi que usin les aplicacions que hem de publicar.

- Servidor d'aplicacions: més endavant tractarem de manera independent aquest component, però és important saber en aquest punt si serà o no necessari usar-lo per al nostre projecte. Segons el servidor d'aplicacions que usem, ens haurem d'assegurar o no que el servidor web triat sigui compatible amb aquest. Hi ha una forta comunicació entre el servidor web i el servidor d'aplicacions; per tant, aquest punt és fonamental per a la nostra elecció.

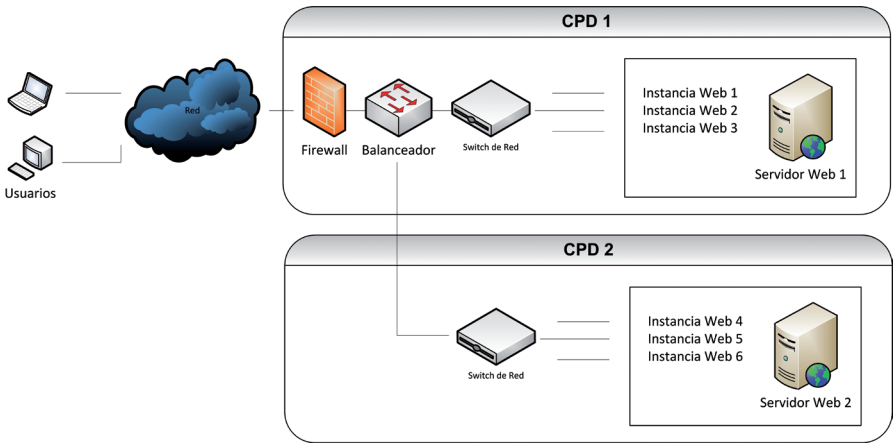
Els punts anteriors ens ajudaran molt a decantar-nos per un servidor web concret, però finalment, l'experiència pròpia, els coneixements de l'equip o els requisits del client marquen d'una manera molt important la nostra decisió final.

Una vegada triat el servidor que usarem, ens hem de plantejar quantes instàncies d'aquest servidor necessitarem. Per definir aquest nombre ens hem de basar en el punt de capacitat i els estudis que hàgim analitzat en aquest. Amb aquestes dades i sabent el nombre d'usuaris concurrents de les nostres aplicacions, i el nombre de peticions mitjanes de cada usuari cap al nostre servidor, podrem fer una estimació de quantes instàncies del nostre servidor haurem de tenir arrencades per complir les necessitats inicials del projecte. Segons la capacitat del maquinari que tinguem disponible, a més a més podrem decidir quantes instàncies hauran de córrer en cadascun dels servidors físics destinats a albergar el servidor web. Un altre punt important per decidir aquest nombre serà el dels requisits de disponibilitat que ens hagin marcat, de manera que podrem optar per aquestes configuracions bàsiques:

- Diverses instàncies del servidor web sobre un únic servidor físic: d'aquesta manera, si una instància deixa de funcionar per algun motiu, les altres seguiran donant servei. Per contra, si el servidor físic falla, perdrem completament el servidor web i estarem en un cas d'indisponibilitat.



- Una o diverses instàncies del servidor web sobre diferents servidors físics: d'aquesta manera evitem el problema anterior, ja que encara que un dels servidors físics deixi de funcionar, l'altre seguirà donant servei. A partir d'aquí, podem analitzar si els servidors físics han d'estar en la mateixa ubicació o separats, fins i tot si han de tenir diferents línies d'alimentació, però tots aquests temes es tractaran més profundament en el pròxim capítol.



Segons el que hem descrit anteriorment, hem de pensar com es connectaran els usuaris al nostre servidor web si aquest té diverses instàncies. Normalment, aquestes instàncies escoltaran en diferents adreces IP o, si no n'hi ha, en diferents ports sobre una mateixa adreça IP. Per evitar tenir diversos dominis i que cadascun apunti a una adreça o a un port determinats, o haver de distribuir diferents adreces a les quals s'han de connectar els usuaris, hem d'usar *balancejadores de càrrega*, el funcionament dels quals ja s'ha estudiat en el tema anterior. El tipus i nombre de balancejadores que usem dependrà del nombre d'instàncies i de connexió que esperem rebre.

Arribats a aquest punt, ja podrem fer una valoració inicial sobre les necessitats corresponents al servidor web que tindrà el nostre disseny.

2.1.1.3. El servidor d'aplicacions: necessitats i disseny

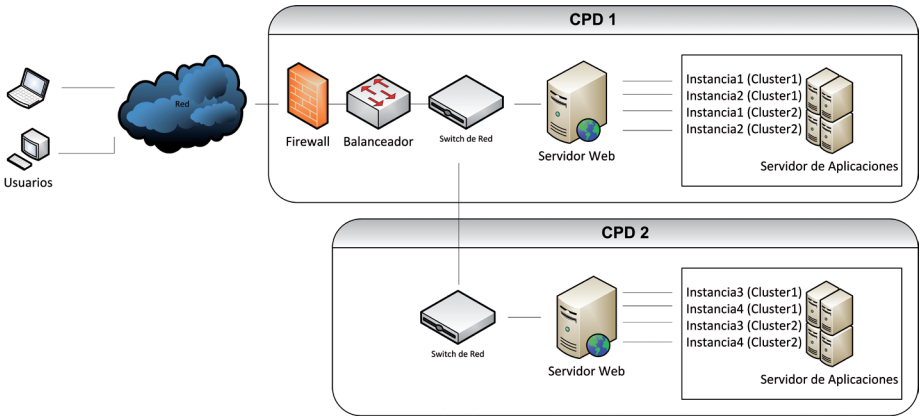
No totes les arquitectures web tenen un servidor d'aplicacions. El seu ús dependrà fonamentalment de les necessitats i el disseny de les aplicacions que hi

hem d'instal·lar i publicar. Hi ha diverses opcions on triar en el mercat, tant del costat comercial com del lliure. Per poder ajudar-nos en l'elecció, de la mateixa manera que amb els servidors web, ens podem fixar en una sèrie de característiques que ens orientaran cap a la millor opció per al nostre disseny segons les nostres necessitats:

- Arquitectura: de la mateixa manera que en els servidors web, un punt decisiu en l'elecció d'un servidor d'aplicacions és l'arquitectura sobre la qual l'instal·larem. No tots els servidors d'aplicacions són compatibles amb totes les arquitectures (per exemple, Internet Information Server o IIS només és compatible amb arquitectures basades en x86 o x64 sobre Windows), i per tant serà fonamental conèixer per endavant aquest punt a l'hora de decantar-nos per alguna de les opcions del mercat.
- Llenguatges de programació: a l'hora de triar el servidor d'aplicacions, és fonamental saber a priori quin llenguatge de programació i quins estàndards usaran les aplicacions que s'executaran sobre aquest. Atès que la tasca fonamental del servidor d'aplicacions és executar-les i proporcionar-los serveis i estàndards de dades i comunicacions, aquestes aplicacions seran les que marcaran d'una manera decisiva la nostra elecció.
- Complexitat i completitud: és important recordar-nos que el nostre disseny, després de la seva implantació, tindrà un manteniment al llarg de la seva vida útil; per tant, hem de tenir en compte la complexitat que per fer aquesta tasca tingui el servidor d'aplicacions que triem. També és important fixar-nos en les característiques i capacitats que ens ofereix el servidor d'aplicacions, com ara els estàndards que implementen, les facilitats d'accés a les diferents fonts de dades que ens proporciona... Ja sigui per requeriment de les aplicacions que anem a instal·lar o per la possibilitat de creixement d'aquestes en el futur, com més complet sigui el producte que triem, menys risc tindrem d'equivocar-nos en un futur (sense oblidar tenir sempre molt en compte la complexitat amb vista al seu manteniment).
- Seguretat: malgrat que tindrem per davant un servidor web i que la comunicació amb el nostre servidor d'aplicacions es farà a través d'aquest, és important fixar-nos en aquest punt per evitar possibles ensurts i atacs.
- Escalabilitat i disponibilitat: els servidors d'aplicacions, en executar localment els processos de les aplicacions, fan un ús intensiu dels recursos del sistema sobre el qual estiguin instal·lats, i per tant és important valo-

rar les possibilitats d'ampliació que tingui aquest i si aquesta es pot fer en «calent», sense necessitat que afecti el servei (per exemple, arrencar una o diverses instàncies noves d'aquest puntualment segons la càrrega de treball). També és important fixar-nos en les opcions que ens ofereix amb vista a la disponibilitat. Hi ha servidors d'aplicacions que ofereixen solucions de clúster, de manera que si perdem una instància, la resta seguiran donant servei i els usuaris connectats a la instància perduda mantindran la seva sessió i serà transferida a les que encara queden operatives.

- Servidor web: tal com vam comentar en la secció de servidors web, és important tenir en compte que siguin compatibles amb el nostre servidor d'aplicacions, i per tant hem de tenir en compte aquest punt en l'elecció de tots dos.
- Rendiment: com hem comentat respecte dels servidors web i com comentarem en tots els components, és important comparar estudis de rendiment i fiabilitat de les diferents opcions que ens trobem en el mercat per tal de triar la que més s'ajusti a les nostres necessitats. Com sempre, a Internet podrem trobar diferents estudis que ens ajudaran en aquesta tasca, encara que és millor evitar les pàgines dels mateixos fabricants dels productes i intentar trobar estudis objectius i lliures de possibles interessos.



En la imatge, dos servidors físics d'aplicacions amb dos clústers, cadascun format per quatre instàncies

Després de fer l'estudi anterior, ja tindrem un servidor d'aplicacions triat per al nostre disseny i, de la mateixa manera que amb el servidor web, ens haurem de plantejar el tipus de disponibilitat que volem per a aquest. Els servidors d'aplicacions també tenen la possibilitat d'aixecar diferents instàncies i, per tant, la casuística que ens trobem és molt semblant a l'apartat anterior. En aquest cas, ens hem de fixar que la majoria d'aplicacions necessiten una sèrie de variables lligades a l'usuari que es connecta, les quals, habitualment, es desen en la *sessió* d'aquest. Per resoldre aquests problemes, els servidors d'aplicacions, en lloc d'aixecar instàncies independents les unes de les altres, tal com hem comentat abans, implementen solucions de clúster per compartir les sessions d'usuaris entre les diferents instàncies. Per tant, hem de tenir present no solament les instàncies que tindrem i en quants servidors físics les arrencarem, sinó també en quants clústers estaran repartides aquestes.

2.1.1.4. La base de dades: necessitats i disseny

Encara que no és un component exclusiu dels portals web, les bases de dades són molt importants per a un gran nombre d'aquests, en especial per als que estan compostos d'aplicacions que emmagatzemen una gran quantitat de dades o en fan un ús intensiu. Podem disposar d'una base de dades connectada directament a les aplicacions que corren sobre un servidor web mitjançant llibreries o connectors propis d'aquestes, o connectades a un servidor d'aplicacions que implementi mètodes o interfícies de connectivitat amb la base de dades triada. De la mateixa manera que en els apartats anteriors, intentarem definir una sèrie de característiques o paràmetres bàsics pels quals guiar-nos a l'hora de decantar-vos per una opció en el nostre disseny:

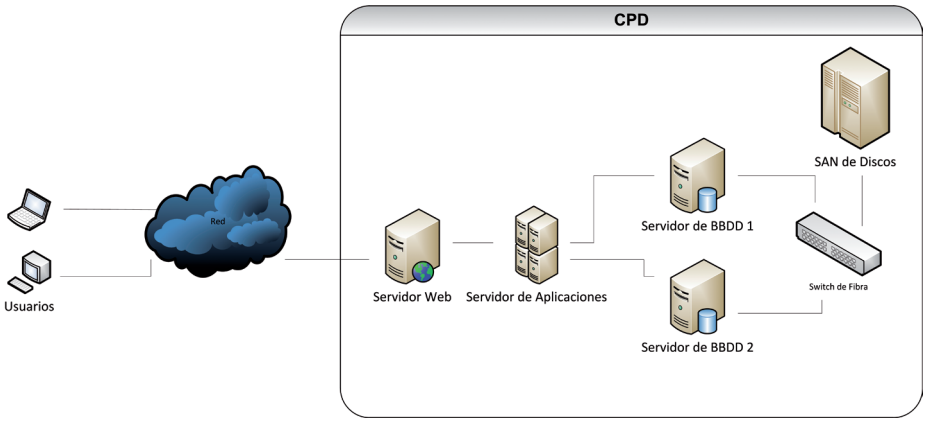
- Volum de dades: segons la quantitat de dades que hem d'emmagatzemar, és important saber com es comporta la base de dades triada amb el volum esmentat. No totes les opcions del mercat estan preparades per albergar un emmagatzematge de dades massiu. També és important tenir en compte el rendiment que ens oferirà la base de dades triada treballant amb un volum de dades determinat. En les especificacions del producte podrem trobar referències a les recomanacions del fabricant respecte als límits d'emmagatzematge recomanats per al seu producte.
- Tipus de dades: segons el tipus d'aplicacions i les necessitats d'aquestes, ens veurem obligats a desar uns tipus de dades o altres (vídeos, música, binaris,

textos...). És important saber el tipus de dades que pot emmagatzemar la base de dades per la qual ens inclinem i com els desa, per tal de saber si és compatible amb les nostres necessitats.

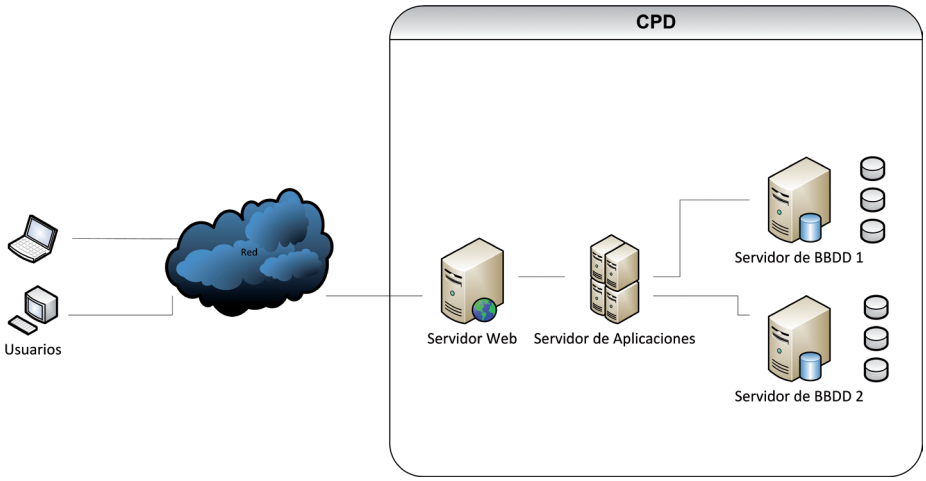
- Rendiment: en moltes aplicacions que fan un ús intensiu de dades o que treballen amb un volum molt gran d'aquestes, un dels principals colls d'ampolla respecte al rendiment i temps de resposta sol ser l'accés a les dades. És important saber com reacciona la nostra base de dades a unes determinades consultes, movent un volum i un tipus de dades determinat. Seria important que coneguéssim aquestes necessitats de les aplicacions abans de decantar-nos per un producte o un altre.
- Escalabilitat: la quantitat de dades és una les coses que més poden canviar al llarg de la vida d'un portal web i, per tant, hem d'estar segurs de com respondrà la nostra base de dades a una variació important d'aquestes. També és possible que el nombre d'aplicacions que ataquen la nostra base de dades pugui variar i augmentar al llarg del temps, i per tant és recomanable que la capacitat de creixement del producte triat sigui ràpida i fàcil.
- Seguretat: la manera correcta per a la implantació d'una base de dades en un portal web és que només el mateix servidor web o el servidor d'aplicació (si n'hi ha) siguin els que s'hi connectin. És important aïllar-la de l'exterior, ja que en molts casos, la informació que hi desem pot ser privada o susceptible de ser atacada per agents externs no desitjats. Malgrat tot, és important garantir que la seguretat que ens aporti el producte triat compleixi totes les necessitats que requereix el projecte. Aquest punt és especialment important en aquells que desin dades acollides a lleis de regulació imposada per la legislació vigent del país en què es dugui a terme la nostra activitat.

Una vegada analitzats tots els punts anteriors i altres que ens imposin des dels requisits del projecte, i després d'haver fet l'elecció del producte més adequat i de la mateixa manera que en els components anteriors, ens haurem de plantejar el tipus de disponibilitat que volem per al producte. En el cas de les bases de dades, també disposem de la possibilitat d'aixecar diferents instàncies d'aquestes i en molts casos implementen solucions de clúster. El problema que ens trobem en aquest cas fa referència a on tenim emmagatzemades les dades. Podem implementar diferents sistemes d'emmagatzematge:

- Emmagatzematge compartit en un element extern als servidors: en aquest cas, tindrem les dades en discos externs als servidors on tinguem les bases de dades, de manera que tots dos puguin accedir-hi. Aquesta topologia ens permet tenir dos tipus de configuració de clúster: actiu/passiu, en què una instància està funcionant usant les dades i l'altra espera que la primera es pari (o falli) per començar a funcionar, o actiu/actiu, en què diverses instàncies estan usant les dades i, si una falla, les altres continuen donant servei. També hem d'estudiar l'opció de tenir dues cabines de discos externes que repliquin les dades entre si de manera automàtica, però no totes les existents en el mercat fan aquesta funció i, per tant, hem de tenir aquest paràmetre en compte a l'hora de decantar-nos per una solució així.

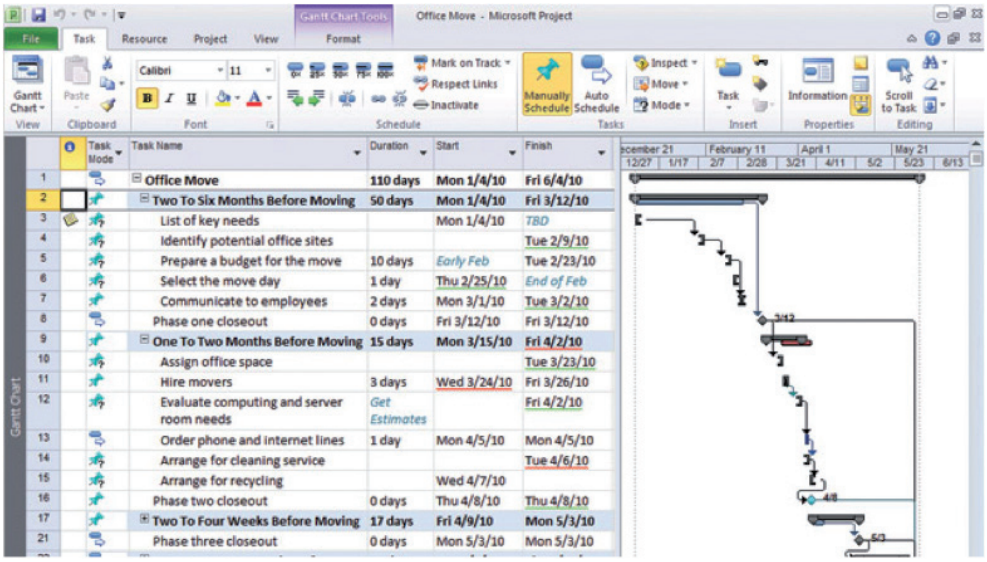


- Emmagatzematge intern o propi de cada servidor: al contrari que en el cas anterior, en aquesta casuística cada servidor disposa del seu propi emmagatzematge, i per tant haurem de buscar mecanismes per replicar i sincronitzar les dades entre els diferents emmagatzematges, de manera que sempre siguin coherents i estiguin actualitzats. En aquest cas només podrem tenir implementacions de clúster del tipus actiu/passiu.



2.1.2. Planificació

En aquest apartat analitzarem els passos de la planificació que cal fer en qualsevol projecte d'implantació de continguts web. Atès que aquest procés depèn de cada tipus de projecte, de la mateixa manera que en els apartats ante-



En la imatge, un exemple de planificació de projecte usant l'eina de Microsoft Project

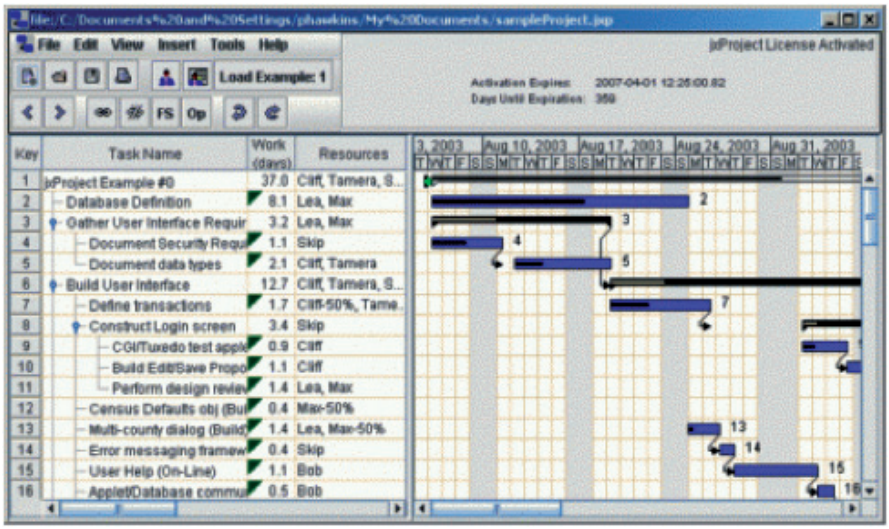
riors, intentarem ser el més generalistes possibles i comprendre els processos més importants d'aquesta fase. No hem d'oblidar que el procés de planificació no és exclusiu de projectes per a continguts web, sinó que és comú a gairebé tots els projectes; per tant, podem trobar en el mercat diverses eines que ens ajudaran a definir i a controlar les diferents etapes del nostre projecte (un exemple d'eina coneguda és Microsoft Project o, en el cas d'una eina gratuïta, Gantter).

Dins de la planificació d'un projecte hi ha dues variables fonamentals:

- El temps: en els inicis del projecte, hem de conèixer els terminis marcats per a aquest. Segons les necessitats que ens plantegin, serà fonamental complir els terminis estipulats o podrem tenir un cert marge de maniobra o desplaçament en els terminis esmentats. La forma més comuna per mesurar el temps en un projecte és traspassar-lo a hores o jornades, segons com vulguem organitzar el treball efectiu. Són més recomanables les hores en projectes que impliquin desenvolupament.
- Els costos (diners): és el factor que influirà directament en la distribució del temps i en gairebé totes les decisions del nostre projecte. A tall d'exemple, si disposem de tres mesos per a la implantació i posada en funcionament d'un contingut web, segons els diners que destinem al projecte, podrem comptar amb més o menys persones per desenvolupar-lo i, per tant, la distribució de tasques i hores variarà substancialment. Hem de tenir en compte que en determinats projectes, el pressupost serà general per tot el que abraci el projecte esmentat (maquinari, programari, recursos humans, treballs...) i ens tocarà a nosaltres la distribució dels recursos monetaris; per tant, la fase de planificació s'ha de fer en un moment inicial del projecte, ja que a nivell de costos pot influir directament en la fase anterior de disseny i en la presa de decisions en aquesta. D'altra banda, en la majoria d'ocasions, serà a nosaltres a qui se'ns demani un pressupost o una estimació monetària d'un projecte determinat, per la qual cosa una petita planificació juntament amb el disseny ens ajudarà a ser més precisos i a no desviar-nos-en en cas que sigui acceptat.

Una vegada disposem de les variables anteriors, gairebé qualsevol procés de planificació i control d'un projecte el podem dividir en les tres fases fonamen-

tals que estudiarem a continuació. Cal tenir present que la segona i tercera fases s'aniran alternant al llarg de la vida del projecte i una complementarà l'altra.



En la imatge, un exemple de planificació de projecte usant l'eina **Gantter**

2.1.2.1. Pla d'acció

Després de tenir clars els components que necessitarem per al nostre projecte, i els possibles desenvolupaments, hem de crear un pla d'acció adequat tant per al desplegament dels components de maquinari, instal·lació i posada en marxa, com per als possibles desenvolupaments i posada en producció dels diferents components de programari. Per comprendre un projecte complet, desenvoluparem un pla d'acció base, en què tocarem tots els possibles punts que ens podrem trobar en una posada en producció i desplegament de components web. No hem d'oblidar que, segons el projecte en què ens trobem, part d'aquestes fases ja estaran fetes o haurem d'alterar l'ordre a causa de problemes o casuístiques concretes del projecte esmentat. Una guia ve bé per no partir de zero, però finalment és el gestor del projecte el que ha de decidir l'ordre i la prioritat de les coses basant-se en les dades de què disposa, les necessitats dels diferents interessats en el projecte i de quan i quants recursos pugui tenir disponibles.

Les fases d'una definició d'un pla d'acció genèric per a l'inici d'un projecte podrien ser les següents:

- **Presentació i acceptació dels requisits del projecte**: la millor manera d'assegurar-nos que des d'un principi totes les parts que intervenen en el projecte estaran d'acord amb el que volem fer és redactar un document amb les necessitats que es pretenen cobrir i amb el que s'espera del projecte que començarem. Aquest document ha de ser acceptat per tots els interessats i ens ajudarà al llarg de la vida del projecte quan ens sorgeixin dubtes o quan hi hagi alguna discrepància respecte del que cal fer.
- **Definició i recerca de recursos**: una vegada sabem el que necessitem fer, hem de definir, d'una manera molt més específica, cada component que necessitarem per portar a terme el treball (tal com s'explica en l'apartat de disseny). Al seu torn, hem de buscar tots els recursos necessaris per al desenvolupament del projecte, tant en el pla humà com d'infraestructures. En aquesta fase, hem de tenir en compte molts factors, per exemple, els recursos humans, i segons la durada del projecte, hem d'estimar quantes hores de treball efectiu ens ofereix una persona (cal tenir en compte vacances, hores de descans…) o en la compra de components de maquinari hem de recordar que es poden produir retards en els lliuraments i, per tant, hem de procurar estimar temps mitjans (encara que un proveïdor en un principi ens doni temps més baixos).
- **Estimació de temps i costos**: arribats a aquest punt, ja podem fer un calendari estimatiu del que durarà el nostre projecte i de les fites de lliurament d'aquest. Com que ja sabem els recursos que necessitarem, hem de calcular els costos de cadascun i tenir en compte una petita desviació a causa de possibles problemes que ens hi puguem trobar. Aquest és el moment en què hem de presentar l'oferta formal al client o als interessats del projecte perquè donin el vistiplau a totes les dades i verificar que el que proposem s'ajusta al document inicial redactat, i també als interessos del client.
- **Desenvolupament i instal·lació**: en aquesta fase és on comença el treball efectiu del projecte. Totes les parts estan d'acord en el que cal fer i han acceptat els costos i temps estimats. Durant aquesta fase, haurem de realitzar les instal·lacions d'infraestructures i productes, i el desenvolupament (si cal) d'aplicacions. No hem de perdre de vista en cap moment el calendari

que hem definit en la fase anterior i ens hem d'anar ajustant als objectius que s'hi defineixen.

- **Posada en marxa del servei**: una vegada conclosa la fase anterior (o no, segons com hàgim definit el projecte, per exemple, podem engegar una part del projecte mentre desenvolupem altres parts d'aquest), passarem a la posada en marxa del servei. Durant aquesta fase, haurem d'haver reservat temps i recursos per a l'anàlisi de possibles errors i per a la seva resolució fins que el servei quedi funcionant correctament i respongui a les necessitats que havíem definit al començament del projecte. En aquest punt, hem de rebre el vistiplau dels interessats inicials.
- **Explotació, millora i monitoratge del servei**: segons com s'hagi definit el projecte, ens tocarà també a nosaltres mantenir el servei que hàgim creat i posat en marxa. Aquest període pot ser indefinit o amb una vida finita i, per tant, pot ser que el projecte tingui una vida útil finita o que s'allargui en el temps fins que deixi de caldre (data indefinida). Els projectes que s'allarguen indefinidament necessiten uns càlculs de costos i recursos continus, i solen tenir formes de mesurar la seva qualitat mitjançant indicadors clars que hem de definir a l'inici del projecte i amb els quals totes les parts hi han d'estar d'acord. A més a més d'assegurar el funcionament correcte del servei, l'haurem d'anar millorant i adaptant en molts casos segons que vagi passant el temps. També serà important definir una estratègia clara de monitoratge i control. Aquesta fase l'analitzarem amb més profunditat en apartats posteriors.
- **Finalització i tancament del projecte**: com hem comentat en la fase anterior, no tots els projectes són finits i, per tant, no tots han d'arribar a aquesta fase. En aquest moment hem d'analitzar totes les fases que hem realitzat, els problemes que ens hem trobat i el grau de satisfacció dels interessats en el projecte. Hem d'elaborar un document de tancament del projecte en què totes les parts han d'estar d'acord en el que s'ha aconseguit i, si n'hi ha, en el que no s'ha aconseguit.

Ara que tenim clares les fases genèriques de qualsevol projecte, ens centrarem en com gestionar correctament les fases de desenvolupament, instal·lació i posada en marxa, ja que són en les que més problemes ens podrem trobar i on una presa de decisions correcta serà fonamental per portar a terme amb èxit el projecte.

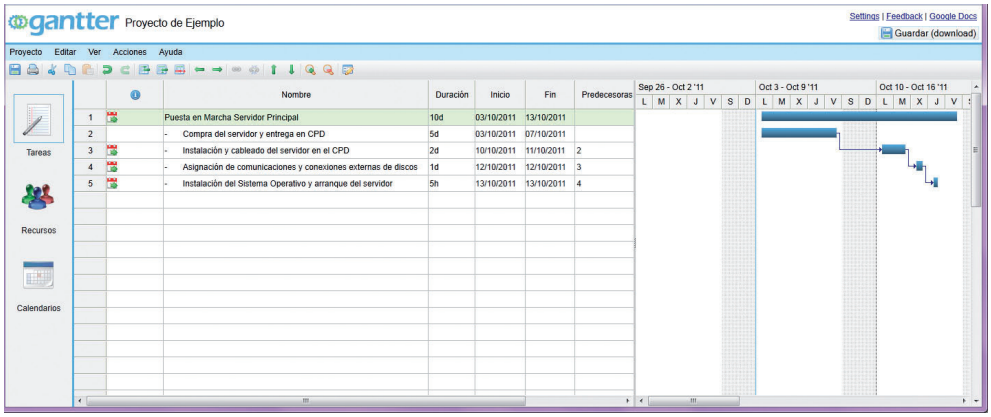
2.1.2.2. Seguiment i control

En qualsevol projecte és fonamental fer un seguiment correcte de les diferents tasques que s'hi faran; així mateix, un control dels temps i costos conscienciós ens ajudarà molt a satisfer les fites marcades i a portar a bon port el projecte.

Per fer-ne un seguiment correcte, ens ajudarà molt que hàgim realitzat correctament les fases de definició i estimació de temps i costos. En aquestes fases, haurem definit les tasques i subtasques necessàries per portar a terme la totalitat del projecte; per tant, podrem agafar com a base aquest document per fer-ne el seguiment. Com més hàgim dividit i identificat les tasques que hàgim de realitzar, més fàcil serà controlar els temps i comprovar la consecució dels objectius.

Podrem entendre millor el seguiment d'una tasca amb l'exemple següent. Suposem per a això la tasca de «Posada en marxa del servidor principal». Aquesta tasca pot comprendre moltes subtasques i, per tant, el millor és dividir-la i estimar el temps que ens portaran aquestes últimes i així tenir l'estimació total de la tasca principal. Una possible divisió seria la següent:

- Compra del servidor i lliurament en CPD (10 dies).
- Instal·lació i cablejat del servidor en el CPD (2 dies).
- Assignació de comunicacions i connexions externes de discos (1 dia).
- Instal·lació del sistema operatiu i arrencada del servidor (5 hores).



En la imatge, la planificació d'aquesta fase del projecte usant l'eina **Gantter**

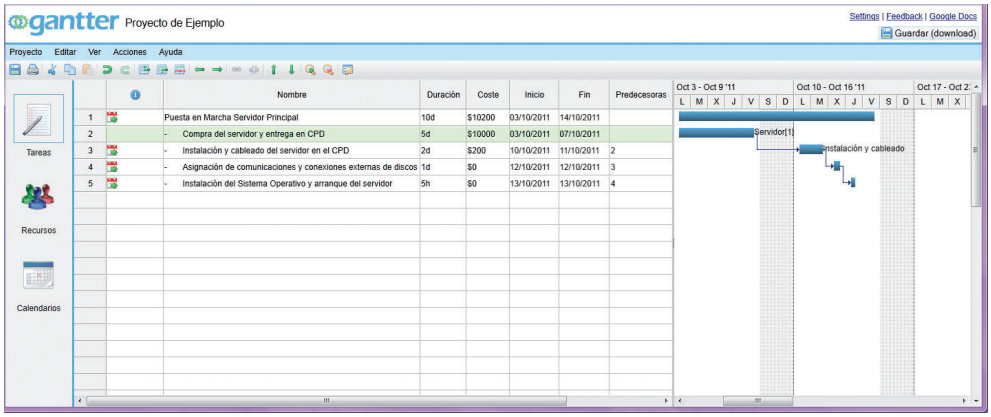
Tenint en compte aquesta divisió, la tasca principal tindria una estimació de 13 dies i 5 hores. Per fer-ne un seguiment correcte, hem de fixar la data d'inici i anar comprovant cada cert període de temps que les subtasques s'estan completant en el temps previst. Ens `podríem marcar com a fites les dates en què haurien de finalitzar cadascuna de les subfases, però això és arriscat, ja que si, quan arriba la fita, fem el seguiment i aquest no s'ha aconseguit, ja no podrem fer res per solucionar-lo.

Com podem veure, no és fàcil definir el moment per fer el seguiment de cada fase, atès que no hi ha un mètode que funcioni sempre, ja que dependrà de la fase i del projecte. Un mètode genèric per fer un seguiment passaria per analitzar cada subfase, el seu temps d'execució i el pes que aquesta té dins el temps total de la fase principal o del projecte. En l'exemple anterior, la primera subfase té un temps estimat de 10 dies i, per tant, un pes molt important dins el temps de la fase principal. Atesa aquesta situació, seria un error esperar fins a la seva finalització per comprovar si s' ha realitzat correctament i el que és més recomanable és fixar-nos punts de control al llarg d'aquesta per comprovar que segueix els cursos adequats, per exemple, cada 2 dies (cada 25% de la fase). Seguint el mateix exemple, si ens fixem en l'última subfase, té un cost en temps de 5 hores i, per tant, molt poc pes dins de la fase principal; en aquest cas, sí que podrem esperar a la seva fita de finalització per comprovar si s'ha realitzat correctament.

De la mateixa manera que fem un seguiment del temps, no ens hem d'oblidar dels costos i, per tant, també hem de fer un control monetari en les diferents tasques del projecte. Normalment, en la mateixa divisió que hem fet per estimar els temps, podem estimar els costos i d'aquesta manera cada subfase tindrà perfectament definits els seus costos. Seguint l'exemple anterior, la divisió podria quedar així:

- Compra del servidor i lliurament en CPD (10.000 €).
- Instal·lació i cablejat del servidor en el CPD (200 €).
- Assignació de comunicacions i connexions externes de discos (0 €).
- Instal·lació del sistema operatiu i arrencada del servidor (0 €).

El que podem observar en primer lloc és que no totes les subtasques han de comportar un cost monetari. En el nostre exemple, en comprar el servidor, ja hem tingut en compte que inclogui la instal·lació d'un sistema operatiu i la configuració de les comunicacions i els discos. Si no és així i s'ha de destinar



En la imatge, la planificació, incloent-hi els costos, d'aquesta fase del projecte usant l'eina **Gantter**

una persona del nostre grup a realitzar aquestes tasques, la forma més fàcil de calcular el cost seria calcular les jornades que dedicarà aquesta persona al treball i quant cal cada jornada d'aquesta persona. Vist aquest exemple, la tasca completa de «Posada en marxa del servidor principal» tindria un cost monetari de 10.200 €.

Vist tot això, podem concloure que un seguiment i control del nostre projecte ens ajudarà a saber on som en cada moment i quant ens falta per acabar cadascuna de les tasques en què hàgim dividit el treball.

2.1.2.3. Gestió de riscos i replanificació

En qualsevol projecte al qual ens enfrontem, hem de tenir en compte que no tot sortirà bé a la primera i que és molt difícil que les nostres previsions es compleixin tal com les hem fet. Per això és fonamental que fem una gestió de riscos correcta i que la nostra planificació inicial sigui adaptable a aquests riscos i, per tant, que accepti una replanificació.

Perquè el treball que realitzarem en aquesta fase sigui més fàcil, és important que en les previsions de costos i temps que hem realitzat en les primeres etapes del projecte hàgim tingut en compte un marge d'error acceptable per a totes les parts, de manera que si ens desviem lleugerament en els temps o en els costos, no es vegi compromès el projecte sencer.

En el tipus de projectes que ens ocupa (instal·lació i posada en marxa de serveis web), moltes fases tindran dependència de les anteriors i, per tant, que una

de les fases s'allargui pot comportar que s'endarrereixin les que depenen d'aquesta. Per posar un exemple, no podrem instal·lar el programari del servidor web o del servidor d'aplicacions fins que el servidor físic no estigui arrencat i s'hagi instal·lat el sistema operatiu.

En la fase de seguiment anirem observant com evoluciona una tasca i, si detectem algun risc en la seva evolució, hem d'engegar el nostre pla de contenció de riscos. Cada problema que ens trobem s'haurà de tractar d'una manera concreta i és difícil donar unes directrius genèriques per abordar les diferents dificultats que ens trobar em al llarg de la vida del nostre projecte. Podem fer una aproximació genèrica basant-nos en els problemes més comuns:

- Orientativament, podem apuntar que, quan ens trobem amb una subtasca que no complirà els temps previstos, hem de replanificar la resta de subtasques de la tasca principal en què ens trobem perquè aquesta no es vegi compromesa. Si fins i tot així no aconseguim complir la fita de finalització a temps de la tasca principal, ens hem de centrar en totes les tasques que en depenguin i intentar rebaixar els temps que hi havíem destinat perquè el projecte sencer no pateixi un retard (aquí és quan ens ajudarà molt haver previst marges d'error en la planificació inicial de temps).
- L'afer es complica més quan fem el mateix control de risc sobre els costos, ja que, normalment, se solen ajustar molt més els projectes en aquest sentit que en el de temps. Quan una subtasca amb un cost estimat supera aquest, hem d'intentar recuperar aquest cost en una altra de les tasques del projecte on hàgim pogut reservar un marge de diners més ampli. No hem d'oblidar que el temps també és diner i que, per tant, si aconseguim reduir jornades de treball d'alguna de les tasques o subtasques, probablement això en redueixi els costos i puguem intentar compensar l'augment en el cost d'una altra de les fases respecte a les estimacions inicials.

En conclusió, aquesta fase de gestió de riscos està molt lligada amb l'anterior de seguiment i anirem alternant constantment entre elles al llarg de la vida del projecte. Si aconseguim gestionar correctament totes les fases descrites, podrem portar gairebé qualsevol projecte a una conclusió satisfactòria per a tots els que hi estiguin interessats.

2.1.3. Instal·lació i desplegament

En aquest apartat ens centrarem en la part d'instal·lació i desplegament de serveis web. Per abordar d'una manera més pràctica aquest apartat, partirem d'un exemple inicial i analitzarem tots els passos que hauríem de fer en un projecte real. Atès que en apartats anteriors hem tractat més profundament les parts de definició i disseny, en l'exemple donarem per finalitzades aquestes fases i ens endinsarem directament en la part de treball sobre maquinari i programari. Com a excepció, analitzarem en alguns punts de la instal·lació diferents productes que actualment es troben en el mercat (recordeu que això ja ho hauríem d'haver fet en la fase de disseny).

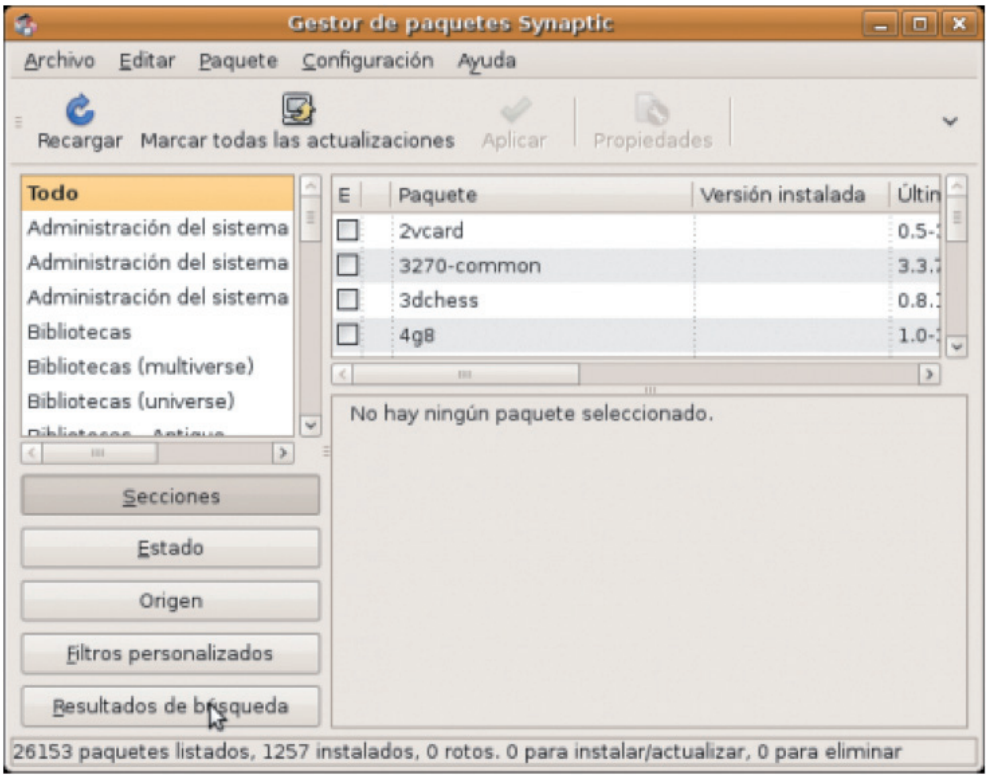
> La universitat ens ha demanat que li proporcionem un entorn web preparat per albergar una sèrie d'aplicacions fetes a Java que implementen una xarxa social per a l'alumnat. A causa dels fons destinats al projecte, ens hem decantat per usar tres servidors virtuals que es troben en el núvol de la universitat. A més a més, un altre dels requisits del projecte és usar el màxim nombre de productes Open Source possibles i evitar els comercials a causa, de nou, de restriccions de pressupost. Atesa l'envergadura i els requisits del projecte, hem decidit que necessitarem dos servidors web, dos servidors d'aplicacions i una base de dades. Per al balanceig de connexions entre els dos servidors web, la mateixa universitat ens ofereix un balancejador de càrrega de maquinari que es troba en la mateixa xarxa on són els servidors. Per tenir una millor redundància del servei i atesa la limitació de servidors, hem decidit que un servidor virtual albergui un servidor web i un altre d'aplicacions, el servidor virtual següent contindrà el mateix i en l'últim instal·larem la base de dades.

Donat l'enunciat, dividirem en tasques els passos següents que farem com si haguéssim definit un pla d'acció però sense aprofundir en temps i en costos com hauríem d'haver fet en el pla esmentat:

Tasca 1. Instal·lació del sistema operatiu i posada en marxa de servidors virtuals. Un dels requisits que tenim per triar el sistema operatiu és que els servidors on l'instal·larem són virtuals i, per tant, hem de comprovar que sigui compatible amb aquests servidors. Per intentar complir el requisit del projecte d'usar el màxim nombre de programari Open Source gratuït, ens decantarem per un sistema Linux (no hem d'oblidar que tenim altres alternatives, com un Oracle Solaris o, en cas de poder comprar productes comercials, tenim també

Microsoft Windows). Un altre requisit que hem de contemplar és que els productes que volem instal·lar sobre aquest sistema operatiu tinguin una versió disponible per a aquest. En el mercat actual tenim moltes distribucions de Linux que compleixen tots aquests requisits, i per tant podem analitzar cadascuna i decantar-nos per la que més ens convenci. Les més famoses en aquests moments en el mercat per usar com a servidors són Red Hat Enterprise Linux i SUSE Linux Enterprise Server, però aquestes dues versions no són gratuïtes, ja que estan destinades a empreses, per la qual cosa si ens volem decantar per complir el requisit de despesa zero, haurem d'anar a distribucions una mica més pròpies d'escriptori però que poden exercir perfectament com a servidors, per exemple openSUSE, Ubuntu o Fedora (n'hi ha moltes altres, però no és el nostre objectiu valorar-les totes). Per al nostre cas ens quedarem amb Ubuntu 11.04 64 bits (atès que la nostra arquitectura és de 64 bits). Els passos d'installació són senzills i només hem de seguir l'assistent que ens guia al llarg dels diferents passos. Recordeu que en aquest moment podem configurar l'adreça IP que se'ns hagi facilitat per poder-nos connectar al servidor. Com que és una distribució per a equips d'escriptori, se'ns instal·larà l'entorn gràfic que encara que no ens fa falta per als propòsits de servidor, no ens molesta i pot ser útil per administrar d'una manera més fàcil el servidor. Cal recordar també que en aquest tipus de distribucions no s'activen els serveis de connexió típics que té un servidor Unix per accedir-hi per xarxa (ssh, telnet, ftp...), per la qual cosa una vegada realitzada la instal·lació hem d'instal·lar-los i activar-los nosaltres mateixos des de les utilitats que ens ofereix el sistema per a això (Gestor de Paquets Synaptic). Una vegada instal·lat i configurat el sistema operatiu, com que estem treballant amb màquines virtuals i probablement qualsevol solució que usem per a això ens proporciona l'opció de clonatge de màquines, el més còmode és fer un clonatge de la ja creada en lloc de tornar a instal·lar altres dues vegades el sistema operatiu. Aquesta opció de clonatge la podem fer en aquest moment o, com que dos dels nostres servidors seran pràcticament iguals a un servidor web i un servidor d'aplicacions, podem endarrerir el clonatge fins haver instal·lat i configurat aquests productes en el servidor que ja tenim muntat, la qual cosa ens estalviarà temps i és una manera de reduir els costos d'aquesta fase. El clonatge, encara que és més fàcil i comú en màquines virtuals, no és una opció exclusiva d'aquestes. En cas de tenir màquines físiques també en podem fer un clonatge, sempre que el maquinari dels dos ens ho permeti (si és idèntic, no hi haurà problema en el clonatge). Per al nostre exemple, el més òptim és fer un clonatge ara, de manera que posteriorment

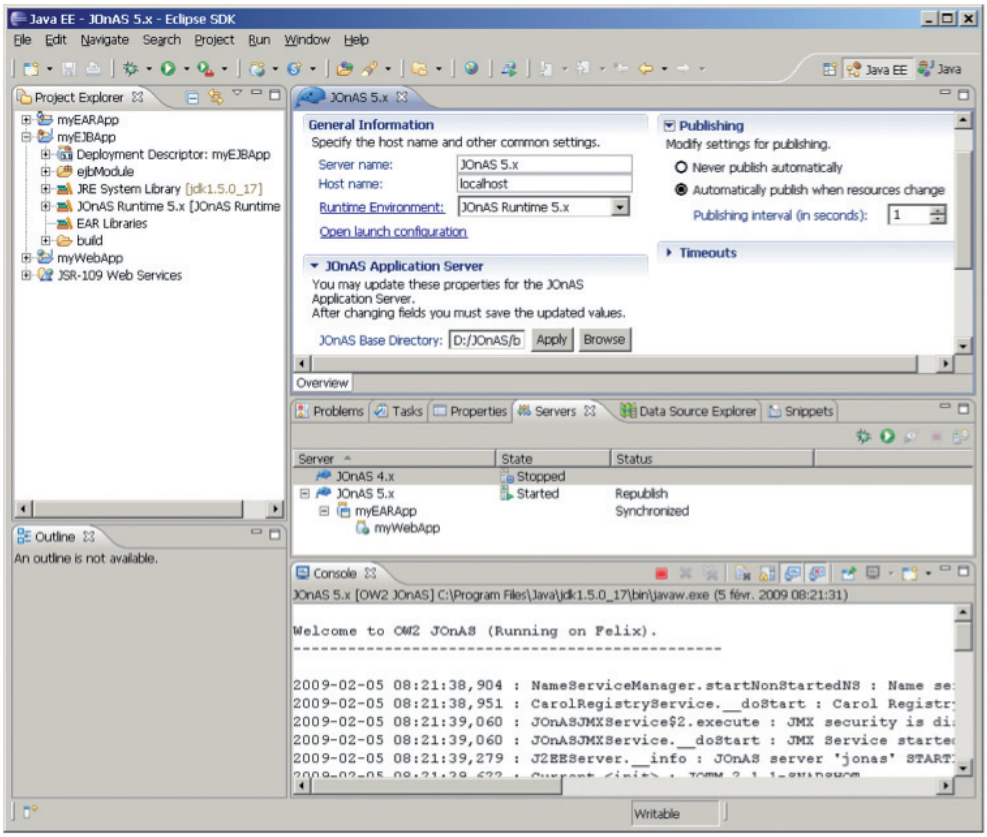
instal·lem la base de dades sobre la màquina que clonem amb el sistema operatiu net i un altre posteriorment quan hàgim instal·lat la resta de servidors. Una vegada realitzat el clonatge, ens hem d'assegurar de configurar l'adreça IP adequada a cadascuna de les màquines perquè no hagi un duplicat d'aquestes a la xarxa i hi puguem accedir sense problemes. Tots aquests processos ens han costat unes tres hores de temps, però no hem d'oblidar que ens queda un clonatge per fer una vegada superades les fases següents, per la qual cosa una valoració inicial correcta en cost de temps per a aquesta fase hauria estat d'unes cinc hores per tal de tenir un petit marge de possibles problemes que ens puguem trobar.



En la imatge, el gestor de paquets Synaptic d'Ubuntu des d'on podrem instal·lar la major part dels components que necessitem

Tasca 2. Instal·lació i configuració del servidor d'aplicacions. Per a l'elecció del nostre servidor, en aquest cas, disposem de dues premisses principals: ha de poder servir aplicacions Java i s'ha de poder instal·lar en el Linux Ubuntu 11.04 que hem triat com a sistema operatiu. En el mercat actual hi ha una àmplia varietat de servidors d'aplicacions. Els més coneguts en ús comercial són IBM WebSphere i Oracle Weblogic. Tots dos estan certificats amb J2EE i són molt estables i usats per moltes empreses importants per la seva robustesa, escalabilitat i prestacions. El nostre projecte ens demanava usar el màxim nombre possible de programari lliure, i per tant ens hem de centrar en aquesta categoria. En el mercat actual podem trobar diversos servidors d'aplicacions lliures i molt potents. Els més coneguts són JOnAS (OW2 Consortium), JBoss (Red Hat) o GlassFish (Oracle), entre altres. Alguns d'aquests últims també tenen versions comercials més econòmiques que les anteriors, encara que menys utilitzades fins ara en grans projectes. Per al nostre, ens quedarem amb JOnAS 5.2.1, ja que compleix tots els nostres requisits i a més a més ens ofereix prestacions com ara la creació de clústers i el balanceig de càrrega entre les diferents instàncies que arrenquem del servidor. Tot això ens ajudarà a configurar un entorn d'alta disponibilitat entre els dos servidors que han de tenir instal·lat el producte. Per fer la instal·lació del servidor, hem de baixar el producte i mirar els requisits d'altres productes que ens indiqui el fabricant que han d'estar instal·lats en el nostre servidor abans de procedir amb la nostra instal·lació. En aquest cas, els més rellevants seran el JRE Enviroment 6 i el paquet ant per fer la configuració base de les instàncies del servidor (entre altres). La instal·lació i configuració completa d'un servidor d'aplicacions és un procés llarg i complex, segons el nivell de seguretat i optimització que es vulgui tenir. Com que no és l'objectiu d'aquest capítol fer una guia d'instal·lació, passarem a resumir la configuració que tindrem al final del procés. En el nostre cas, necessitarem instal·lar una instància mestra en un dels servidor. És un requisit del servidor d'aplicacions que per cada grup de servidors físics (o virtuals en el nostre cas) hi hagi un instància d'aquest tipus. A més a més d'aquesta instància, crearem altres dues instàncies normals, que seran les que realment serviran les nostres aplicacions i formaran part del clúster (no oblidem que aquest servidor l'hem de clonar i que, per tant, finalment tindrem quatre instàncies dins del clúster i la instància mestra només en un dels servidors). Quan tinguem les instàncies creades, hem de configurar un clúster amb el nom que decidim i hem de fer membres d'aquest a les instàncies que hàgim creat. A més a més, haurem de modificar el tipus de ba-

lanceig de càrrega que vulguem entre les diferents instàncies si decidim no utilitzar el que ens ofereix per defecte el producte. Després d'aquests treballs, haurem de configurar la seguretat del servidor i aplicar les configuracions que se'ns hagin facilitat des de la part que desenvoluparà les aplicacions, perquè aquestes funcionin correctament en el nostre servidor. Probablement tindrem una IP diferent per a les instàncies que la inicial que hem configurat en el servidor, per tant hem de configurar-la usant diferents ports per a cada una de les instàncies. Quan instal·lem el servidor web en la pròxima fase, farem el clonatge del servidor que hem configurat i una vegada arrencat el nou haurem de modificar el nom de les instàncies i IP per adaptar-les a les que realment han d'anar en aquest servidor, de manera que no siguin les mateixes que teníem en l'origen. Després de tot això, serà el moment de configurar el clúster correcta-



En la imatge, la consola de JOnAS

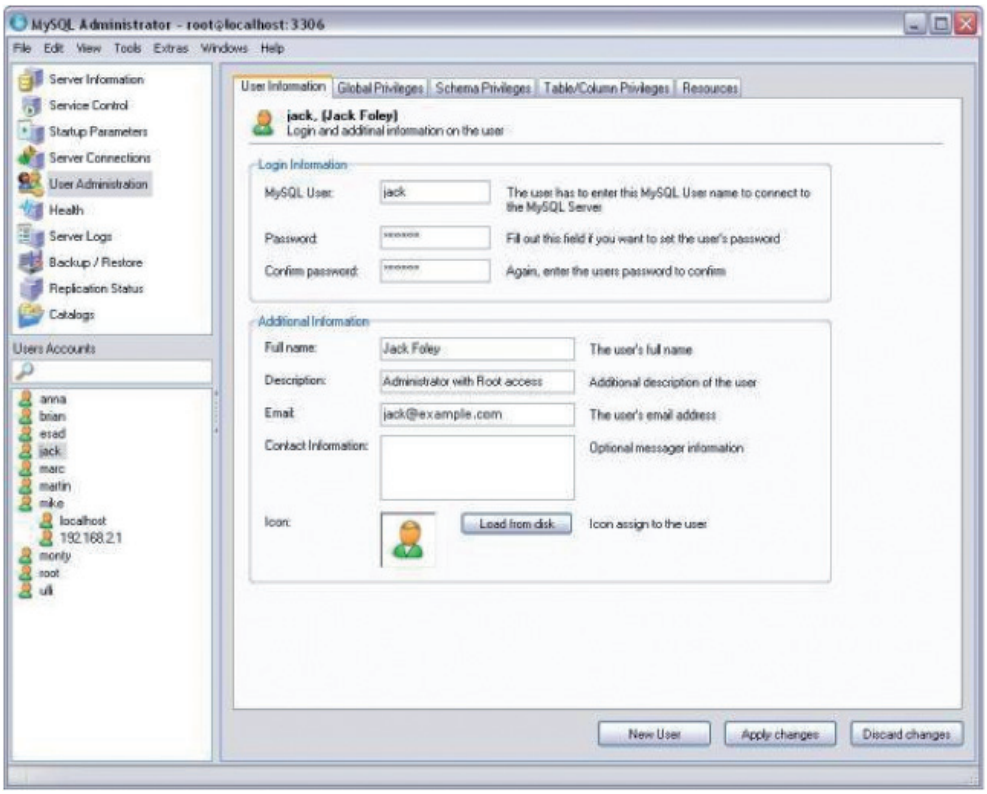
ment amb els dos servidors i arrencar en primer lloc la instància mestra i després totes les instàncies que compondran el nostre servei. Atesos els passos i la seva complexitat, podem estimar que aquesta fase tindrà un cost temporal d'entre un i dos dies de treball (ens podem assegurar i posar-ne dos per tal de tenir un marge de temps en cas que ens trobem amb problemes no previstos en el nostre disseny inicial).

Tasca 3. Instal·lació i configuració del servidor web. Els requisits d'aquesta fase tornen a ser els mateixos que en l'anterior. Només que en aquest cas també haurem de tenir present el servidor d'aplicacions que hem triat, ja que hem de poder dirigir les peticions que es facin al nostre servidor web cap a aquest i, per tant, això sigui potser el que més ens limiti. En aquest cas, el mercat està molt més definit i l'elecció serà fàcil. El servidor web per excel·lència i el més conegut és Apache, compatible amb la majoria de sistemes operatius i configurable amb tots els servidors d'aplicacions que hem esmentat en la tasca anterior. Altres productes que podem trobar actualment en el mercat són Microsoft Internet Information Server (IIS) d'ús comercial, compatible només amb sistemes Windows i més limitat respecte als servidors d'aplicacions amb els quals pot interactuar. En la part de codi obert també trobem Cherokee, menys conegut que Apache i més difícil de connectar a servidors d'aplicacions. Vist tot això, la nostra opció és clara: instal·larem Apache HTTP Server 2.2, per a la qual cosa podem baixar el paquet des de la pàgina web del fabricant i instal·lar-lo manualment, o aprofitar que és un paquet inclòs en el gestor de paquets d'Ubuntu i instal·lar-lo des d'aquest. Una vegada tinguem instal·lat el servidor, l'haurem de configurar assignant-li la IP que tinguem destinada a aquest efecte i fent que es carregui la llibreria del nostre servidor d'aplicacions per poder connectar-nos amb ell (el nom de la llibreria ens l'indica el fabricant i es descriu en els manuals d'instal·lació). Aquesta llibreria sol tenir el seu propi fitxer de configuració per poder definir el clúster o les instàncies del servidor d'aplicacions al qual ens hem de connectar; per tant, no hem d'oblidar de configurar-la correctament. També és el moment ara d'instal·lar els certificats de seguretat necessaris si tenim intenció d'habilitar i configurar l'SSL en el nostre servidor web per a un accés més segur. Ara sí que és quan durem a terme el clonatge del servidor i modificarem la configuració tant del sistema com del servidor d'aplicacions i el servidor web per adaptar-lo als paràmetres que tinguem destinats per al servidor esmentat. No hem d'oblidar facilitar als encarregats del balancejador de

càrrega les IP que hàgim configurat en els nostres servidors perquè puguin donar-nos al seu torn la IP del balancejador o el DNS que digui a aquestes on s'han de connectar els usuaris per accedir als nostres serveis web. Aquesta fase té poca complexitat i, per tant, la podem valorar en unes dues o tres hores de treball (ja que els temps de clonatge i configuració del sistema i servidor d'aplicacions entren dins dels costos temporals de les seves fases corresponents).

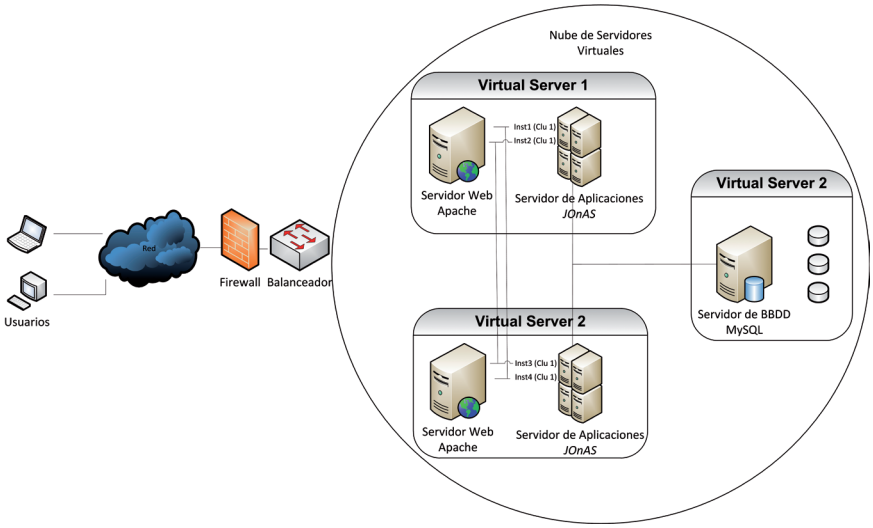
Tasca 4. Instal·lació i configuració del servidor de base de dades. El mercat a què es refereix la base de dades també es troba força definit i podem destacar les opcions següents que s'adaptin als nostres requisits tant de sistema operatiu com connectivitat amb el servidor d'aplicacions. En l'àmbit comercial destaca Oracle Database 11g (la seva actual versió), que és la més usada juntament amb les seves versions anteriors per la majoria d'empreses. També podem trobar IBM DB2 9.7, com la competència més directa de l'anterior, i pel que fa al sistema operatiu Windows, tenim Microsoft SQLServer. Tots ells són servidors de bases de dades molt potents i poden ser usats per a qualsevol projecte, però tots tenen elevats costos de llicències i suport, per la qual cosa cal tenir molt present aquest punt a l'hora de triar-los. En el terreny que ens ocupa per al nostre projecte, programari lliure, podem trobar com a opcions destacades MySQL i PostgreSQL, que també són servidors de bases de dades potents i podran satisfer pràcticament qualsevol necessitat que ens trobem en els projectes que vulguem implementar. Ens decantarem per MySQL 5.1, ja que, a més a més, de ser prou potent per als requisits que ens han demanat els desenvolupadors de les aplicacions, es pot instal·lar sense problemes en el sistema operatiu que hem triat i el seu connector JDBC es pot usar sense problemes amb el nostre servidor d'aplicacions. En aquest cas, de la mateixa manera que en el servidor web, podem triar baixar-nos el paquet d'instal·lació i instal·lar-lo de manera manual o usar el gestor de paquets que ens ofereix Ubuntu, que ens facilita la tasca. Una vegada instal·lat el servidor, ens hem de descarregar i instal·lar les eines administratives per tal de poder configurar i gestionar d'una manera més senzilla el nostre servidor. La configuració bàsica que hi hem d'aplicar consisteix en la creació d'una nova instància de base de dades i dels usuaris que necessitem per accedir-hi, instància que haurem de configurar segons les necessitats que hagin de tenir les aplicacions i reservar l'espai necessari per al seu creixement. El nostre servidor de base de dades, de la mateixa

manera que els anteriors, també tindrà una adreça IP associada que haurem de configurar. Finalment, hem de connectar el servidor d'aplicacions a la base de dades que hem creat mitjançant la instal·lació del connector de base de dades en els servidors on tenim instal·lats els servidors d'aplicacions i la posterior configuració d'aquests. Ateses les tasques d'aquesta fase i els possibles problemes que puguin aparèixer a l'hora de les connexions, podem reservar entre cinc i vuit hores per a la seva realització.



En la imatge, la consola d'administració de MySQL

Després de la finalització de les tasques, ja tenim desplegat el nostre servidor de continguts web llest per a la instal·lació i publicació de les aplicacions que ens faciliti la universitat. Podem resumir el que hem desplegat i les seves característiques principals de la manera següent:



En la imatge, el diagrama de com quedarà el nostre sistema

Dos servidors web amb balanceig de càrrega entre ells i alta disponibilitat, ja que es troben en servidors diferents; al seu torn, aquests estan connectats a un clúster de servidor d'aplicacions amb quatre instàncies que balancegen la càrrega entre si i que, de la mateixa manera, ofereixen alta disponibilitat pel fet de trobar-se distribuïdes entre diferents servidors. Finalment, tenim un servidor de base de dades que, en aquest cas, serà el punt més vulnerable del sistema, ja que tan sols té una instància i, si tenim algun problema amb aquesta, deixarem de donar servei a totes les aplicacions que requereixin de la base de dades per funcionar. Amb aquesta infraestructura serem capaços de servir continguts web bàsics instal·lats directament en els servidors web o aplicacions complexes que compleixin els estàndards J2EE, les quals estaran instal·lades en el nostre servidor d'aplicacions; al seu torn, disposem d'una base de dades relacional per poder emmagatzemar una quantitat important de dades, i ampliar així el ventall de possibilitats de les nostres aplicacions. Dit tot això, podem concretar del nostre projecte que és d'una mida mitjana i compleix la major part del que es necessita actualment al mercat i que, per descomptat, compleix les expectatives que ens va demanar la universitat en el nostre exemple.

Continuant en la línia anterior, proposarem dos nous exemples de diferent envergadura per veure com abordar diferents mides i necessitats de projectes. En el primer, ens enfrontarem a un projecte per a una gran empresa amb uns

requisits exigents en tots els aspectes i amb un pressupost d'acord amb la magnitud del projecte. En el segon, plantejarem un projecte per a una petita empresa amb recursos limitats i amb unes necessitats molt més reduïdes i quotidianes.

Exemple 1: Air Telecom ens ha demanat la creació d'un portal corporatiu on oferiran la seva imatge de marca, i un servei de venda i contractació en línia de tots els seus productes. Com a dades de referència ens han facilitat informació del seu lloc web actual, el qual rep aproximadament unes 200.000 visites al dia amb un volum de contractació i venda d'aproximadament 1.500 operacions diàries. També ens han facilitat informació sobre les dades que tenen actualment i les aplicacions que necessiten desplegar en el servei web. Les dades de clients i operacions que necessiten les aplicacions són aproximadament de 15 Tb i les aplicacions que necessiten penjar en el lloc estan basades en Java i compleixen els estàndards J2EE. Al seu torn, el servei ha de residir en les instal·lacions que l'empresa té per a la seva infraestructura d'IT, consistent en dos CPD col·locats en plaques tectòniques diferents i connectats mitjançant una macrolan de fibra tant per a la xarxa de comunicacions com per a la xarxa de dades. Tots dos CPD disposen d'una cabina SAN de dades de la marca EMC, model Symmetrix, en què ens poden reservar 20 Tb d'espai per al nostre projecte. Per descomptat, com a requisits del projecte, hem de complir una alta disponibilitat total del servei amb redundància total de tots els components. Disposem d'un pressupost inicial d'1.000.000 €.

Seguint amb el que hem anat aprenent fins ara, hem de triar tots els components que han de formar el nostre sistema; en aquest cas, des de la part de maquinari fins als productes de programari que l'han de compondre. Anirem pas a pas desenvolupant el projecte:

- Ateses les necessitats plantejades pel client, hem de definir una infraestructura redundant en tots els seus components. A més a més, hem de tenir en compte el volum de visites que tindrà el nostre sistema, i la seva càrrega de treball. Tenint en compte tots aquests factors i basant-nos en l'experiència i la informació que puguem trobar, ens decantarem per muntar 16 instàncies web segures (instàncies https, ja que, com que es tracta d'un servei web on es pot comprar i contractar productes, comportarà la introducció de dades personals i confidencials per part dels usuaris) distribuïdes en dife-

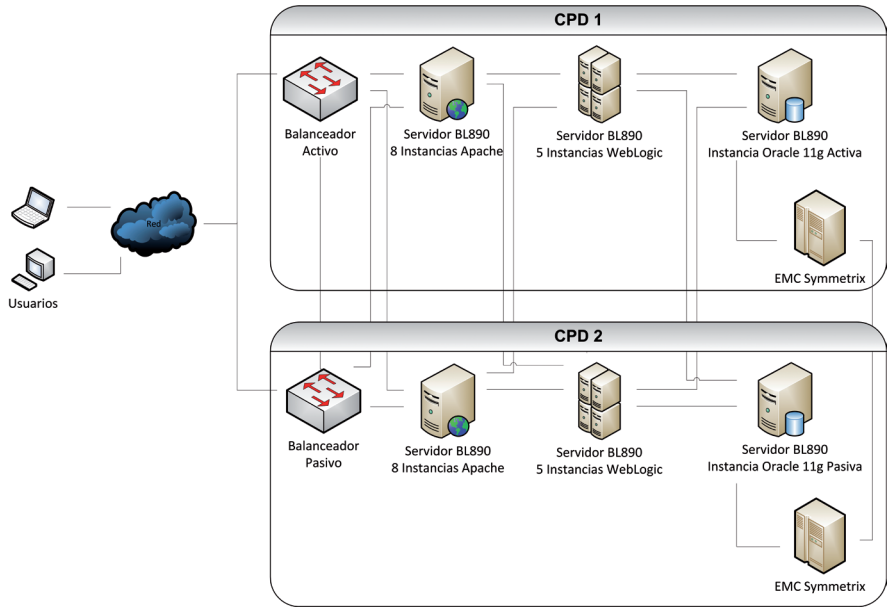
rents servidors ubicats en els dos CPD. Per a les aplicacions, instal·larem un servidor d'aplicacions que implementi un clúster amb deu instàncies actives també distribuïdes entre els servidors ubicats en els dos CPD. Per a la base de dades, hem decidit tenir dues instàncies amb un clúster actiu/passiu, de manera que cadascuna doni arrencada a cadascun dels dos CPD. Finalment, respecte a les dades, les tindrem en les cabines SAN que ens facilita el client, de manera que estaran actives en una d'elles, que s'encarregarà de replicar-les a l'altra síncronament (hem comprovat prèviament que el model de cabina que té el client pot fer aquesta operativa; altrament, l'haurem de fer mitjançant programari i, per tant, ens haurem de preocupar que el producte que muntem de base de dades o la solució clúster el puguin dur a terme). Amb totes aquestes mesures garantim que, encara que un CPD sencer es quedi inoperatiu, nosaltres podrem continuar prestant el servei per l'altre amb una afectació mínima en el servei.

- Ara arriba el moment de definir l'arquitectura de maquinari que albergarà el nostre servei i que es pugui adaptar al disseny que hem definit en el punt anterior. Com hem comentat anteriorment, el mercat ens ofereix moltes opcions en l'àmbit del maquinari i ens hem de basar en molts factors per triar-ne una opció o l'altra. Moltes vegades en grans projectes les necessitats es treuen a concurs perquè els proveïdors de maquinari ens aportin solucions i competeixin amb els preus; d'aquesta manera, podem maximitzar l'estalvi en la compra de la solució definitiva. Per a aquest exemple i per a poder analitzar productes menys freqüents en petits projectes, ens decantarem per una solució UNIX que ens ofereix HP. Comprarem sis servidors HP BL890 amb 16 cores Itanium*2* i 256Gb de memòria RAM cadascun. Dos els dedicarem als servidors web, uns altres dos als servidors d'aplicacions i els dos últims al servidor de base de dades. Un de cada tipus l'ubicarem en el CPD 1 i els altres tres en el CPD 2. El sistema operatiu que anirà sobre aquests servidors serà el HP-UX 11.31, que és un sistema basat en UNIX de 64 bits, que reconeix correctament els processadors Itanium2.
- Atès que tindrem diverses instàncies de servidors web i, tal com hem comentat anteriorment, necessitarem tenir un balancejador que reparteixi la càrrega entre totes les instàncies. A més a més, tenim el requisit de total disponibilitat; per tant, hem de comprar dos balancejadors i col·locar-los en CPD diferents, de manera que, si un falla, l'altre assumeixi el servei. En

aquest cas, ens decantarem per la solució Enterprise de Loadbalancer, ja que ens ofereix dos balancejadors de càrrega en alta disponibilitat amb les prestacions de connexions simultànies que compleixen de sobres les nostres necessitats de projecte i a un preu força competitiu. El client ens facilitarà els *switches*, tant de xarxa com de SAN, que necessitem, ja que els té disponibles en les seves instal·lacions. Si no fos així, hauríem de tenir present que també hem d'adquirir aquests components.

- Per al servidor web, de la mateixa manera que en el cas anterior, ens decantarem per Apache HTTP Server 2.2 per a HPUX. Arrencarem 8 instàncies d'aquest en cadascun dels BL890 que tenim en tots dos CPD. Cadascuna de les instàncies tindrà una IP pròpia, que configurarem posteriorment en els balancejadors que hem adquirit. Ara no entrarem en la fase d'instal·lació, ja que és similar a l'exemple anterior.
- En el cas del servidor d'aplicacions, ens decantarem per Oracle Weblogic Server 11g Release 1. A més a més de tractar-se d'un servidor molt estable i d'eficàcia provada, com que, posteriorment, en l'apartat de base de dades, també ens decantarem per una solució Oracle, el fet de comprar diversos components al mateix subministrador ens pot ajudar a rebre descomptes, i així estalviarem en costos. Haurem d'instal·lar el servidor en els dos BL890 destinats a aquest efecte i configurar un clúster amb 5 instàncies en cadascun dels servidors que, com ja sabem, estan en CPD diferents. Cada instància l'arrencarem en un port diferent sobre la mateixa IP (5 en una IP i 5 en una altra, ja que estaran corrent en servidors físics diferents). Aquestes IP i ports les hem de parametritzar en el fitxer de configuració per a Apache que ens proporciona el fabricant. D'aquesta manera, les diferents instàncies web mitjançant una llibreria balancejaran les peticions entre totes les instàncies de servidor d'aplicacions que componen el nostre clúster. En cas de caiguda d'alguna o d'algunes de les instàncies, la mateixa llibreria ho detecta i envia les peticions a la resta d'instàncies fins que aquesta o aquestes tornen a estar actives. No hem d'oblidar que la instal·lació i la configuració d'un servidor d'aplicacions pot ser una operació força complexa i, per tant, haurem de contractar aquest servei al subministrador si no disposem del personal adequat en el nostre equip per fer-ho.
- En el cas de la base de dades, ens inclinarem per instal·lar el servidor Oracle Database 11g Enterprise Edition. Aquest servidor ens garanteix poder mou-

re una gran quantitat d'informació d'una manera eficient (recordem que estem parlant d'un volum considerable de dades, superior o igual a 15 Tb) i que, a més a més, pot rebre un gran nombre de connexions de les nostres instàncies de servidor d'aplicacions. Atès que hem de muntar una solució de clúster amb una instància activa i una altra de passiva, el mateix Oracle ens ofereix la solució Oracle RAC 11g clúster. Aquesta solució s'encarregarà de monitorar l'estat de la instància i, en cas d'error d'aquesta, d'arrencar la instància passiva perquè atengui les peticions. Podríem optar també per una solució activa/activa, ja que aquesta solució d'Oracle ens proporciona una forma de compartició de dades correcta entre les diferents instàncies. De la mateixa manera que en el cas anterior, la instal·lació, configuració i posada en marxa d'un Oracle RAC és una cosa complexa i, per tant, també ho hem de tenir present a l'hora de contractar aquest servei al subministrador en cas de no disposar del personal qualificat per fer-ho.



Finalment, hem construït una solució totalment redundant, en què, si falla qualsevol dels components, això no afectarà el servei, ja que tindrà un suport que seguirà proporcionant-lo. Podríem haver optat per tenir dos servidors de cada

tipus en cadascun dels CPD i repartir el servei entre ells. Aquesta solució també hauria estat vàlida, i fins i tot més segura que la proposada, però també els costos haurien augmentat, si bé es podria compensar a mitges aquesta pujada triant servidors amb menys capacitat de càlcul, ja que tindríem més quantitat d'aquests. Com hem dit, hi ha múltiples solucions per a un mateix problema i serà l'experiència els que ens ajudarà a millorar-les i a decantar-nos per una o per una altra.

Exemple 2: Una petita llibreria vol obrir el seu negoci a Internet i, per tant, vol muntar una pàgina web on els possibles clients puguin veure els llibres dels què aquesta disposa i fer reserves i comandes dels llibres esmentats. La llibreria ja ha comprat l'aplicació que està basada en php i implementa un sistema de pagament segur per Internet. A més a més, per tal de funcionar correctament, l'aplicació necessita una base de dades per emmagatzemar-ne la informació. Entre imatges, contingut i les dades de la base de dades, l'aplicació té un pes inicial d'uns 200 Mb, però pot anar creixent a mesura que el client introdueixi nous llibres a la venda. Com que es tracta d'un negoci petit, no s'espera una gran quantitat de visites ni de trànsit en un inici, per la qual cosa aquestes necessitats són força baixes. El client ja ha fet una inversió important en l'aplicació, i per tant vol invertir el mínim possible en la infraestructura per albergar-la; per contra, tampoc no té una gran necessitats d'alta disponibilitat.

Aquest cas és molt més senzill que l'anterior, encara que també molt més limitat pel que fa a costos. Haurem de buscar, doncs, la millor solució per complir les necessitats del projecte amb una inversió inicial mínima. Com en el cas anterior, desplegarem per parts la solució:

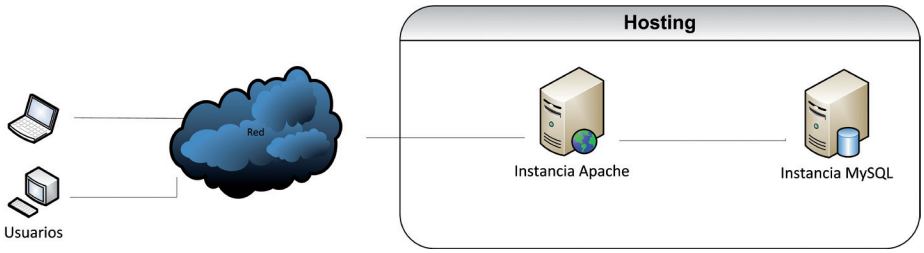
- Com que no tenim un requisit d'alta disponibilitat, que la quantitat de visites inicials serà baixa i que l'aplicació està feta en PHP i, per tant, no requereix un servidor d'aplicacions però per contra sí que requereix albergar dades en un servidor de base de dades, proposarem la solució següent: muntarem una instància web amb les corresponents llibreries per servir pàgines PHP, la qual estarà connectada a una instància de base de dades.
- Com que es tracta d'un client petit, gairebé segur que no té una infraestructura pròpia d'IT, per la qual cosa, en aquest cas, no és recomanable la compra d'un maquinari dedicat per al desplegament del nostre servei. Optarem

per una solució de *hosting* que cobreixi les nostres necessitats. En el mercat hi ha molts proveïdors de *hosting*. Varien amb molta freqüència i, per tant, no en podem nomenar algun en concret. Quan vulguem oferir una solució així, el millor és consultar a Internet com està el mercat en aquell moment concret i decantar-nos per una de les solucions que més garanties ens ofereixi i que millor s'adapti al nostre pressupost. Avui dia, un *hosting* per a un servei web amb una necessitat d'espai de 500 Mb i una transferència mensual de dades de 5 GB/mes pot costar aproximadament de 100 € a 300 € l'any. Per al nostre projecte hem de trobar un *hosting* que ens permeti albergar uns 200 Mb, amb possibilitat de créixer, que el servidor web que puguem instal·lar-hi pugui oferir aplicacions PHP i que ens permeti instal·lar un servidor de base de dades. No oblidem que també haurem de contractar un domini per al servei que volem muntar, per la qual cosa si el servei de *hosting*, a més a més, ens proporciona un domini, ja tindrem solucionat el problema. Com a exercici, podeu buscar a Google «Hosting PHP MySQL» i comprovar el gran ventall de possibilitats que teniu per a un projecte d'aquest tipus.

- Per al servidor web que usarem, hem de tenir present que ens ofereix el *hosting* que haguem triat. La majoria d'ells ens oferiran Apache, per la qual cosa el millor serà decantar-nos per aquesta solució que, com ja hem dit múltiples vegades, està molt ben valorada en el mercat actual. De tota manera, si ens decantem per un *hosting* de servidors Windows, pot ser que ens ofereixin com a solució IIS (Internet Information Server), que també es pot adaptar a les nostres necessitats. No hem d'oblidar que la nostra aplicació està feta en PHP, per la qual cosa el servidor pel qual ens decantem ha de tenir instal·lats i configurats els complements i les llibreries necessaris per poder servir pàgines d'aquest tipus. Tant Apache com IIS disposen d'aquesta funcionalitat. En general, molts *hosting* ja ens proporcionaran un entorn de treball amb el servidor web preinstal·lat i configurat per poder usar pàgines PHP, per la qual cosa si hem triat aquest tipus de *hosting*, és un treball que ja tindrem fet.
- En el cas del servidor de base de dades i atès que es tracta d'una petita quantitat d'informació, ens podem decantar per una solució MySQL o per una PostgreSQL. Totes dues solucions són bones i compleixen de sobres les necessitats que tenim per al nostre projecte, per la qual cosa ens hem de fixar en quina d'elles ens permet albergar el nostre *hosting*. De la mateixa manera

que en el cas del servidor web, molts *hostings* ja ens oferiran la base de dades preinstal·lada i probablement configurada la connexió entre el nostre servidor web i les llibreries PHP amb aquesta. En cas de no ser així, haurem de fer nosaltres aquests dos passos i després d'això haurem de crear la nostra base de dades i importar les dades que ens hagi facilitat el client per començar a treballar. En els dos casos (MySQL i PostgreSQL), disposem d'eines que es poden baixar des de la pàgina del fabricant per fer la importació de dades i creació i configuració de la base de dades.

- Finalment, ens quedarà pujar els continguts que ens hagi proporcionat el client al servidor web que tenim en *hosting* per començar a donar servei. La majoria de *hostings* ofereixen connexions ftp o scp per pujar i manejar contingut.



Aquest exemple és dels més senzills que podrem trobar en el mercat, però també dels més freqüents. Tenen costos molt reduïts i la seva posada en marxa és ràpida i no comporta una gran inversió ni de diners ni de temps. Encara que hàgim optat per una solució de *hosting* en un projecte senzill com aquest, no hem de pensar que aquestes solucions serveixen només per a aquest tipus de projecte. Al mercat hi ha una gran oferta de *hosting* que ens poden garantir alta disponibilitat, i respondre a totes les necessitats que ens pugui plantejar un gran projecte. Això sí, el seu preu s'incrementarà proporcionalment a les nostres necessitats. Independentment de la complexitat o els requeriments del nostre projecte, les solucions de *hosting* estan especialment indicades per a casos com aquest, en què el client no disposa d'una infraestructura pròpia d'IT i no vol invertir en una.

2.1.4. Gestió de la configuració

Fins ara hem parlat de tota l'arquitectura i del programari base que componen una estructura preparada per servir continguts web. En aquest apartat ens centrarem més en les aplicacions i els components que podem servir des de la nostra infraestructura i com controlar aquests continguts.

La gestió de la configuració és el que ens ajudarà a regular i assegurar la validesa de les aplicacions i els continguts que instal·larem i servirem des del nostre servidor de continguts web. És molt important portar un control exhaustiu dels productes que instal·lem en la nostra infraestructura, ja que una aplicació poc provada o amb un funcionament anòmal ens pot portar a una pèrdua de rendiment important o fins i tot a una pèrdua total del servei, no solament d'aquesta aplicació, sinó de tots els components que estiguem servint. Aquesta gestió ens serà especialment útil en projectes de llarga durada, en què no solament hàgim d'engegar la infraestructura, sinó també donar-hi suport i mantenir-la al llarg del temps. L'equip de desenvolupament dels continguts és el responsable principal de portar a terme la gestió de la configuració, de manera que faciliti les dades d'aquesta a l'equip de suport de les aplicacions esmetnades i de les infraestructures que les serveixen. Les dades imprescindibles que ha de portar qualsevol control dels components de programari haurien de ser les següents:

- **Nom del component**: identificador únic per a la nostra aplicació o component dins del sistema.
- **Versió del component**: números i lletres que identifiquin d'una manera inequívoca uns lliuraments del producte d'altres.
- **Estat del component**: l'estat actual en què es troba el nostre component (alfa, beta, producció…).

Amb un control adequat, cada vegada que posem en servei una nova versió del producte, sempre tindrem la possibilitat de tornar enrere i retrocedir fins a la versió que teníem instal·lada prèviament. Aquest comportament ens garanteix una estabilitat en el nostre sistema i és important que es tingui en compte per a qualsevol nova instal·lació. També ens serà útil conèixer aquesta informació a tall d'inventari, tenint sempre una idea precisa del que estem servint des del nostre sistema.

Control de canvis aplicació 1		
Versió	**Estat**	**Data d'instal·lació**
1.0	Producció	14/10/2010
1.5	Producció	18/11/2010
2.0	Producció	05/02/2011
2.1	Producció	20/03/2011
2.5	Producció	08/06/2011
2.1	Producció	09/06/2011
2.8	Producció	21/06/2011

En la imatge, un control de canvis corrresponent a una aplicació exemple

Amb vista als productes de tercers que tinguem instal·lats en el nostre sistema, també podem fer una gestió de la configuració sobre aquests. És important saber les versions que tenim instal·lades d'aplicacions com el servidor web o la base de dades, i a més a més hem de conèixer els possibles pedaços que installem sobre aquests per tal de corregir problemes i veure sempre la compatibilitat que hi ha entre les diferents versions dels productes. Aquest punt és especialment important quan tenim un servidor d'aplicacions interactuant amb un servidor web, ja que, en variar de versió un d'ells, ens podem trobar amb problemes.

Després del que hem vist, podem concloure que la gestió de la configuració es pot aplicar a qualsevol sistema de la informació i encara que té més sentit en entorns de desenvolupament i desplegament d'aplicacions. Ens pot ser molt útil per a entorns d'explotació que utilitzin productes de tercers.

2.2. Qualitat del servei web

Si en l'apartat anterior hem analitzat com dissenyar i instal·lar tota la infraestructura necessària per donar un servei web, en aquest ens centrarem en com analitzar la qualitat del servei que estem donant. Per fer aquesta tasca, en el mercat s'han fixat uns estàndards i unes eines, amb les quals podem mesurar de manera quantitativa fins a quin punt estem donant un servei correcte, tant pel que fa a disponibilitat al llarg del temps com a l'eficàcia amb què

l'estem donant. No hem de perdre de vista que aquestes mesures seran fonamentals a l'hora de demostrar a qui ens hagi encarregat el projecte que aquest s'està servint correctament, per la qual cosa totes les parts han d'estar d'acord en com cal mesurar i qualificar totes les dades de qualitat que s'han d'analitzar. A més a més, basant-nos en les dades que obtinguem en el procés, podrem prendre accions correctores per millorar les dades, i per tant el nostre servei, per la qual cosa el procés de mesurament de la qualitat del servei serà continu al llarg de la vida del nostre projecte des del moment de la seva posada en marxa.

En els apartats següents ens centrarem en els mesuraments més comuns que se solen fer en qualsevol servei web per comprovar-ne el funcionament correcte. Aquests mesuraments, els podem dividir en:

- Disponibilitat del servei
- Acord de nivell del servei
- Estadístiques d'accés al servei

Finalment, veurem com intentar millorar totes les dades que hàgim analitzat anteriorment per tal d'intentar millorar el servei que prestem.

2.2.1. Disponibilitat i acord de nivell de servei

Com ja hem comentat anteriorment, és fonamental a l'hora d'iniciar un projecte posar-se d'acord amb els interessats sobre què volen o les expectatives que tenen sobre el servei final que esperen rebre. Això era important per poder fer un disseny correcte de la infraestructura i una posada en marxa del servei en els terminis fixats. Ara analitzarem com aquesta definició inicial ha de contenir també un acord de nivell de servei per saber clarament que s'espera del nostre servei web una vegada estigui en marxa, ja que això ens influirà no solament en el dia a dia del manteniment del servei, sinó també en com dissenyar-lo abans de la seva posada en marxa perquè finalment pugui respectar l'acord esmentat.

Podem definir l'acord de nivell de servei (també conegut com a SLA, de les seves sigles en anglès Service Level Agreement) com un contracte o document redactat i negociat pel proveïdor d'un servei i el client o els interessats en aquest,

en què es fixa el nivell de qualitat acordat per aquest. En l'acord esmentat s 'han d'establir prioritats, mesuraments i llindars per determinar si el servei s'està prestant correctament o no i les penalitzacions (si n'hi ha) al proveïdor en cas que no compleixi els mínims de qualitat fixats.

Els acords de nivell de servei no són exclusius dels projecte de serveis web i avui dia s'apliquen pràcticament en qualsevol camp per millorar la qualitat dels serveis prestats d'un proveïdor als seus clients. En el cas que ens ocupa a nosaltres, els serveis web, hi ha un tipus de mesuraments establertes i acceptats per pràcticament tots els interessats, que ens ajuden a mesurar la qualitat dels serveis i d'aquesta manera poder plasmar els mínims desitjats en el nostre acord inicial:

- Disponibilitat: quan parlem d'aquest concepte, ens referim al temps al llarg de la vida del servei en què aquest està disponible o presta servei als seus usuaris. Segons la criticitat del servei, el mesurament es pot fer en unitats de temps més o menys grans, per la qual cosa els intervals en què comprovem la disponibilitat seran més o menys prolongats en el temps. La majoria de serveis web, com que estan destinats a usuaris finals, no es poden permetre estar indisposats períodes curts de temps, ja que els usuaris ho notarien, per la qual cosa els mesuraments se solen fer en segons i els intervals de mesurament no solen superar els cinc o deu segons. Cal fixar també cada quant temps s'han de presentar les dades a estudi, tant per la nostra part com per part del client o els interessats del servei. Normalment, i segons la criticitat, l'estudi se sol fer cada dia, setmana o mes. Per expressar els resultats d'una manera més fàcil de comprendre per totes les parts, aquest se sol indicar com un percentatge, de manera que els llindars de servei mínim s'expressin igual i no cal quedar per sota de la cota fixada. Haurem de fixar-nos de nou en la criticitat del servei i negociar amb els interessats els percentatges mínims que hem d'establir, ja que no hem d'oblidar que en molts casos se'ns podran imposar penalitzacions si no hi complim. Aquests percentatges poden ser molt restrictius, i segons de la inversió que es realitzi en el projecte, solen ser més elevats, fins a assolir cotes del 99%.

Control de disponibilitat del servei web 2011								
Contingut	**Gener**	**Febrer**	**Març**	**Abril**	**Maig**	**Juny**	**Juliol**	**Agost**
Aplicació 1	100%	99%	100%	98%	80%	99%	98%	100%
Contingut 1	100%	100%	99%	100%	89%	100%	100%	98%
Aplicació 2	100%	99%	100%	100%	88%	100%	98%	90%
Aplicació 3	99%	100%	85%	100%	80%	100%	99%	100%
Contingut 2	100%	100%	99%	90%	85%	100%	100%	100%
Aplicació 4	99%	88%	100%	100%	90%	97%	100%	98%
Aplicació 5	99%	100%	88%	100%	91%	100%	100%	100%

En la imatge, un exemple de control de la disponibilitat d'un servei web per mes amb un compromiís del 98%

- Rendiment i temps de resposta: aquests paràmetres són els encarregats de mesurar directament la qualitat del servei que oferim als usuaris. El rendiment dels nostres serveis ens ajudarà a determinar com es comporten en diferents circumstàncies de càrrega de treball o connexions d'usuaris concurrents. El temps de resposta determinarà el retard que experimenten els usuaris des que inicien una acció fins que el nostre servei els torna la resposta o el resultat d'aquesta. Per determinar aquests temps, solem tenir autòmats arrencats constantment fent operacions sobre les aplicacions i les pàgines que servim a manera d'usuari, mesurant els temps que triga cada operació. Puntualment, també se solen fer proves d'estrès sobre el nostre servei, la qual cosa consisteix a atacar la nostra pàgina o aplicació amb múltiples d'aquests autòmats i comprovar com es comporta i com varien els temps de resposta. Aquesta última prova també ens ajudarà a determinar la càrrega de treball que pot suportar el nostre servei (nombre d'usuaris concurrents accedint-hi). Per marcar en un acord de nivell de servei els marges que podem suportar de rendiment, hem de tenir en compte tots els factors que hi influeixen, i per tant hem d'especificar que s'espera un temps de resposta màxim de x mil·lisegons per a una determinada operació amb una càrrega de treball de y usuaris concurrents. No hem d'oblidar que en els temps de resposta influeix, d'una manera determinant, el tipus de connexió que hi hagi entre l'usuari i el servidor on estigui allotjat el nostre servei, per la qual cosa si nosaltres no garantirem o no hem de garantir la qualitat i velocitat de la connexió, ha

de constar en l'acord inicial per eximir-nos de responsabilitats en cas que l'excessiu temps de resposta sigui a causa de factors de la connexió esmentada.

Control de temps d'accés mitjans del servei web 2011								
Contingut	**Gener**	**Febrer**	**Març**	**Abril**	**Maig**	**Juny**	**Juliol**	**Agost**
Aplicació 1	0,5s	0,4s	0,5s	0,8s	0,7s	0,3s	0,3s	0,4s
Contingut 1	1,3s	0,9s	0,8s	1,1s	0,6s	0,8s	0,9s	1,3s
Aplicació 2	0,4s	0,6s	1,5s	2,4s	0,9s	0,5s	0,7s	0,4s
Aplicació 3	0,6s	0,7s	0,6s	0,5s	1,3s	0,9s	1,1s	0,5s
Contingut 2	0,7s	0,8s	0,6s	0,8s	0,8s	0,9s	2,3s	1,5s
Aplicació 4	0,6s	2,5s	0,8s	0,9s	1,5s	0,8s	1,8s	0,9s
Aplicació 5	1,8s	0,5s	0,5s	0,8s	1,7s	0,9s	0,5s	0,5s

En la imatge, un exemple de control de temps de resposta d'un servei web per mes amb un compromís de l'1 segon

Introduïts aquests dos factors fonamentals per a qualsevol acord de nivell de servei, podem concloure que en l'acord hem d'establir el percentatge de disponibilitat que ha de tenir el nostre servei, i els temps de resposta i càrrega d'usuaris màxims que aquest suportarà. No oblidem, tal com comentàvem a l'inici d'aquest apartat, que el no-compliment d'aquest acord pot derivar en repercussions de caràcter econòmic o en la pèrdua total del contracte; per tant, hem de ser molt meticulosos a l'hora d'establir els valors que s'han d'assolir i en quines condicions es donaran aquests valors, i descartar responsabilitats que no siguin nostres d'una manera explícita i sense deixar lloc a dubtes posat , en cas que aquestes s'arribin a produir (recordem l'exemple de la connexió). Els valors que puguem reflectir en l'acord estaran molt lligats als recursos de què disposem per fer el nostre projecte, per la qual cosa normalment els projectes amb un pressupost més elevat també solen ser molt més exigents respecte als nivells de servei que s'han d'assolir amb aquests, i no hem d'oblidar, quan el dissenyem en l'etapa inicial, que arribarà un punt que ens hem de comprometre a complir uns acords determinats, i que, per tant, hem de pensar-hi sempre abans de prendre qualsevol decisió definitiva de disseny.

2.2.1.1. Estadístiques d'accés als continguts

Seguint en la línia de mesurar la qualitat i el nivell de servei dels nostres serveis web, hi ha eines al mercat que ens seran molt útils per aconseguir estadístiques, tant de temps com d'accessos als continguts que tinguem publicats. Amb aquestes estadístiques, podrem completar les dades necessàries per als informes que hem de lliurar periòdicament segons que hàgim acordat en el nostre acord de nivell de servei. Els programes d'estadístiques no solament ens ajudaran a l'hora d'establir estadístiques del rendiment i la disponibilitat del nostre servei web, sinó que també podran ser usats pels diferents clients o departaments que tinguin interessos en els continguts que servim des d'aquest. Aquests interessats podran accedir a valuoses estadístiques sobre el contingut més visitat o sobre com ha funcionat amb vista als usuaris un nou contingut que hem publicat...

En la imatge, un exemple de gràfic d'estadístiques de visites d'un lloc web fet amb Google Analytics

Dins el mercat actual el programa o conjunt d'accessoris més conegut i gratuït que ens ajudarà a recopilar tota la informació que necessitem, és Google Analytics. Aquest servei està destinat a recollir estadístiques de qualsevol lloc web, no solament per a tècnics o administradors de webs, sinó, com esmentàvem anteriorment, també molt útils per a departaments de màrqueting o responsables de negocis que basin part de la seva estratègia en els continguts que nosaltres estem servint o que hem ajudat a servir. El funcionament del servei és senzill, però comporta la inclusió d'un cert codi en les pàgines web o continguts que vulguem analitzar. Les dades que recullin aquests petits *scripts* es desaran associades al compte que prèviament hàgim configurat en els servidors de Google i les podrem consultar amb un programa que ens podrem baixar des del lloc del fabricant esmentat. En l'aplicació esmentada tindrem diverses utilitats, com ara gràfics, per poder analitzar les dades d'una manera còmoda per a qualsevol usuari, encara que aquest no sigui un expert.

A més a més d'usar eines ja existents en el mercat, sempre ens podrem decantar per desenvolupar un conjunt concret d'aplicacions i *scripts* que es dediquin a recollir estadístiques i mesurar temps d'accés segons paràmetres millor ajustats a les nostres necessitats. Aquesta opció pot resultar més precisa per a determinats casos, però comporta un cost que hem de valorar i estar disposats a assumir dins del disseny inicial del projecte.

No oblidem en cap moment que si des d'un principi es marca la necessitat de signar un acord de nivell de servei, caldrà poder mesurar el que en especifiquem en aquest acord, i per tant haurem de disposar de les eines oportunes per a això, ja siguin les disponibles al mercat o les desenvolupades per nosaltres mateixos.

2.2.2. Ajust del servei

Després d'aclarir les eines i les formes que tenim de mesurar el nostre servei, hem de saber que tant l'acord de nivell de servei com el servei en si no sempre són elements estàtics i poden variar i se'n poden fer ajustos. Aquests ajustos poden venir motivats per moltes raons, com canvis en les necessitats del client, modificacions en les infraestructures o variacions en la càrrega d'usuaris o peticions que pugui tenir el nostre servei i que en un principi es valoraven inferiors o superiors. Principalment, els ajustos es basaran en les dades estadístiques o me-

suraments que fem sobre el nostre servei, els quals propiciaran que aquest millori i s'adeqüi més al que n'esperem.

Per tot això, quan parlem d'ajust del servei, ens referim als canvis, les millores o les adaptacions que realitzarem en el nostre servei web al llarg de la seva vida per millorar-lo, adaptar-lo o adequar-lo més a les necessitats que se'ns marquin, ja sigui al començament o posteriorment, durant la seva explotació.

No podem preveure o detallar tots els ajustos que es poden fer sobre un servei web, ja que estaran molt lligats al tipus de servei, a com aquest s'exploti, però podem detallar els grups d'ajustos més freqüents i els que se solen fer en pràcticament tots els serveis en explotació al llarg de la seva vida útil:

- Ajustos de maquinari: aquestes són les modificacions o millores que realitzarem sobre la infraestructura de maquinari en la qual corre el nostre servei. Pot venir motivat per millores en el rendiment o la disponibilitat del servei, o per un augment de la capacitat o renovació per obsolescència.
- Ajustos de configuració: són els canvis que realitzem sobre la configuració dels elements de programari que serveixen els nostres continguts, com ara el servidor web, la base de dades, el servidor d'aplicacions... Normalment estan promoguts per *tuning* o posada al punt dels servidors. Es pretén adequar la seva configuració perquè sigui òptima a l'hora de complir les nostres necessitats. De la mateixa manera que en el cas anterior, els canvis de capacitat o disponibilitat també ens portaran a fer ajustos en la configuració del nostre programari base.
- Ajustos dels continguts: amb aquests ajustos ens referim als canvis o modificacions que es realitzen sobre els continguts o programes que servim des del nostre servei web. No hem d'oblidar mai que per molt bé que dimensionem o muntem una infraestructura per donar un servei, gran part del bon funcionament amb vista als usuaris o clients d'aquest serà responsabilitat de les aplicacions que servim. Moltes vegades s'hi han de fer ajustos a manera correctiva o de millora per aconseguir un augment de rendiment.

A més a més dels ajustos que realitzem en el nostre servei, es pot donar el cas que, després de la posada en explotació del nostre servei i analitzant les dades estadístiques, observem que fins i tot fent totes les millores oportunes no arriba-

rem a complir de cap manera l'acord de nivell de servei definit inicialment. Encara que no és desitjable ni habitual, es pot donar el cas que hàgim de redefinir un nou acord més adaptat a la realitat que observem en el nostre servei. Aquesta última opció no sol ser habitual i, per tant, no ens l'hem de plantejar fins que no hi hagi cap altra solució, principalment perquè en el nostre acord estan implicades les parts interessades en el projecte i no és agradable discutir per què no es poden arribar a complir els objectius marcats i pels quals en un principi han pagat.

2.3. Manteniment

Durant aquest apartat estudiarem totes les tasques que haurem de portar a terme durant la fase d'explotació o producció d'un servei web. Com ja hem esmentat anteriorment, en un projecte l'objectiu del qual sigui la posada en marxa i suport d'un servei web, la seva etapa de l'explotació serà la més llarga i en la qual haurem d'analitzar i resoldre els problemes del dia a dia. En els diferents subapartats veurem les tasques més comunes que haurem de fer perquè un servei web es mantingui en producció i, al seu torn, també ens pararem a analitzar els diferents mètodes de supervisió del nostre servei.

Les tasques de manteniment d'un servei web varien segons el tipus de servei, encara que la majoria tenen parts comunes i són en les que ens centrarem més profundament.

Abans d'aprofundir en les tasques, hem d'esmentar que hi ha diferents tipus de manteniment i suport. Segons les dimensions del servei o serveis web que hàgim de mantenir, es poden donar casuístiques en què nosaltres només siguem responsables d'una part del manteniment i altres empreses o departaments s'encarreguin de la resta del manteniment. Per contra, es podria donar el cas que nosaltres hàgim de donar manteniment a totes les parts que componen el servei, des de la infraestructura o el maquinari fins a les aplicacions que servim. En cas que el manteniment no sigui completament nostre, haurà de quedar totalment delimitada la part de la qual ens encarreguem nosaltres i on s'acaben les nostres competències. Això ens estalviarà problemes amb la resta de grups implicats i facilitarà la recerca de responsables, en cas que hi hagi un problema.

Si recordem el que esmentem en la fase de disseny del servei, en l'etapa de

manteniment intervenen principalment uns recursos humans que haurem d'haver tingut presents a l'hora de calcular costos i d'haver dimensionat correctament per poder fer totes les tasques que detallarem a continuació.

2.3.1. Tasques habituals

Dins el manteniment d'un servei web, hi ha treballs quotidians que es realitzen de manera freqüent i són els que tractarem en els pròxims apartats. No aprofundirem en tots, ja que en la seva majoria es tractaran d'una forma més exhaustiva en el capítol de seguretat.

2.3.1.1. Control d'accessos

Com a responsables del manteniment dels serveis web, hem de gestionar d'una manera adequada i molt controlada els usuaris i tipus d'usuaris que poden accedir a aquests serveis. En pràcticament qualsevol entorn d'aquest tipus ens trobarem tres tipus molt diferenciats d'usuaris:

- Administradors: són els que poden fer qualsevol canvi en els components que formen el nostre servei web. Són els últims responsables de la gestió de la configuració en els servidors i els que tenen tots els privilegis necessaris per poder fer qualsevol acció requerida pel servei. Dins d'aquest grup i en instal·lacions grans o gestionades per diverses empreses, es poden fer subgrups, segons les parts del servei a les quals hagi de tenir accés l'administrador. D'aquesta manera podem trobar administradors de sistemes, administradors del servidor d'aplicacions, administradors de bases de dades... Cadascun d'aquests subgrups tindrà només privilegis sobre la part del servei que els afecti, de manera que no puguin modificar components dels quals no siguin responsables.
- Operadors: són els encarregats de fer les tasques quotidianes i de manteniment del servei. Tenen certs privilegis sobre els components del servei, però sense arribar al nivell dels administradors. Normalment, aquests privilegis consisteixen a poder analitzar els *logs* dels servidors, fer algunes modificacions rutinàries sobre aquests i, de vegades, poder parar i arrencar components o parts de components concretes. De la mateixa manera que en el cas

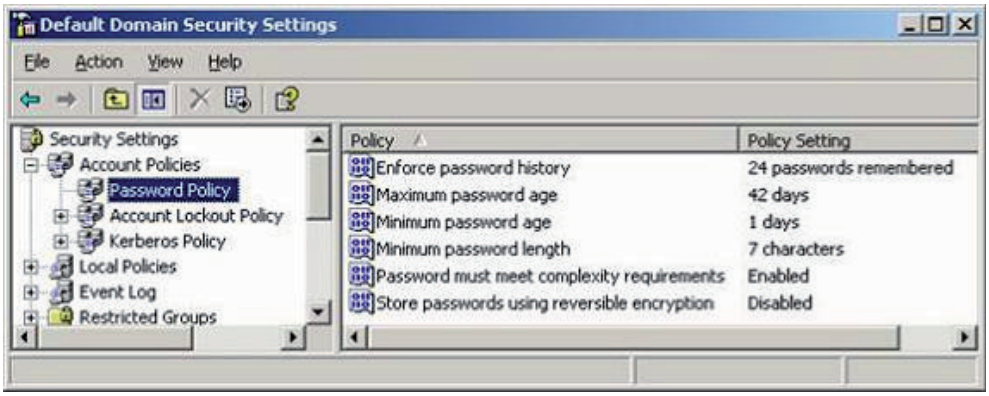
anterior, també ens podem trobar diferents subgrups si dividim el servei en les seves diferents parts fonamentals.

- Usuaris: dins d'aquest grup englobem les persones que accediran als continguts que ofereix el nostre servei. És pràcticament segur que ens trobarem altres subgrups, segons els diferents rols que pugui exercir cada usuari en un contingut concret o diferenciant a quins d'aquests pot accedir cada usuari.

En la imatge, una finestra de control d'usuaris de Windows

Dins de les nostres tasques en el control d'accessos, hi figuren les de creació, esborrament i modificació d'usuaris. També haurem de tenir uns registres d'accessos clars per saber quan i què ha fet cada usuari que accedeixi al nostre siste-

ma. Haurem de definir polítiques de contrasenyes que siguin segures i que ens garanteixin que compleix la legislació vigent, en cas que part dels continguts que oferim en el nostre servei web s'hagin d'atenir a la normativa esmentada. No hem d'oblidar que un control correcte d'accessos i seguiment del que cada usuari realitzi sobre el nostre sistema ens ajudarà tant a protegir el nostre servei com a investigar qui ha fet què, arribat el cas d'haver de fer-ho.



En la imatge, definició de polítiques de contrasenyes per a un domini de Windows

2.3.1.2. Còpies de seguretat

Atès que aquest concepte es tractarà molt més profundament en el pròxim capítol, ens limitarem a comentar que part de la nostra tasca quotidiana en el manteniment d'un sistema de serveis web és la realització de còpies de seguretat. Serà molt important tenir clar el que cal copiar i que tinguem una versió de les còpies que anem fent per poder recuperar la que ens interessi segons els canvis que hàgim anat fent en el sistema. Dependrà molt de cada sistema allò del que ens interessa tenir còpies de seguretat, però podem detallar una sèrie d'elements generals que convé tenir desats en qualsevol sistema:

- Sistema operatiu: és important tenir una còpia de seguretat dels sistemes de fitxers que componen el nostre sistema operatiu. Aquesta còpia l'hem de fer amb una freqüència suficient perquè els canvis que puguem realitzar sobre la configuració del sistema no es produeixin més freqüentment que aquestes i, per tant, perdem versions. En aquest cas, podria ser recomanable fer

còpies incrementals (s'explicaran més endavant) i així estalviar-nos copiar sistemes de fitxers que no varien al llarg del temps, però que sí que ocupen molt espai. Dins d'aquest grup de còpies podem incloure també els fitxers que desen la informació sobre els usuaris que accedeixen al nostre sistema i servei; en aquest cas, la freqüència de còpia ha de ser freqüent, ja que els usuaris poden modificar les seves contrasenyes en qualsevol moment i els hem d'ocasionar el menor trastorn possible en cas d'haver de fer una recuperació de la còpia.

- Fitxers de configuració del servidor web: es pot tenir una còpia de seguretat programada cada *x* dies d'aquests o fer una còpia manual cada vegada que s'hagi de fer una modificació sobre aquests. Encara que no fem modificacions, és important tenir-ne sempre almenys una còpia, per si de cas perdem els que tenim en producció.
- Fitxers de configuració del servidor d'aplicacions: la freqüència de còpia d'aquests és igual que en el cas dels anteriors i hem de tenir en compte les mateixes casuístiques.
- Fitxers de configuració de la base de dades: en cas de comptar amb aquesta, hem de tenir en compte que cal desar els seus fitxers de configuració igual que en els dos casos anteriors.
- Dades de les bases de dades: si optem per aquesta solució, moltes de les aplicacions o continguts que servim desaran informació a la base de dades; per tant, és important fer còpies de seguretat periòdiques i contínues d'aquesta. Els diferents servidors que hi ha al mercat ens ofereixen solucions pròpies per fer les còpies; així, depèn del producte triat, haurem de configurar correctament les còpies segons les especificacions del fabricant.
- *Logs* del sistema i servei (servidors): els *logs* són fitxers que varien i creixen ràpidament, per la qual cosa ens interessa una freqüència de còpia alta. Atès que poden arribar a ocupar molt espai en els nostres servidors, és important treure'n còpies i eliminar-los del sistema principal. Si tenim prou versions de les còpies, podrem consultar el que ha passat en el nostre sistema o servei en qualsevol moment donat i això pot ser molt important per trobar problemes del sistema o investigar possibles atacs o vulnerabilitats en la seguretat.
- Aplicacions i continguts: en aquest cas hem de saber si volem copiar tant els binaris d'aquestes com els fitxers de configuració, per tal de tenir una eersió completa de les aplicacions o, per contra, els binaris ja els gestionem amb el

control de versions i només ens interessa la configuració. L'estudi de la freqüència de les còpies el podem fer extensible a la mateixa casuística que els fitxers de configuració dels altres servidors. En el cas dels binaris, la freqüència correcta seria cada vegada que els modifiquin o els substitueixin dins del servei. No oblidem que, encara que parlem de binaris, dins d'aquest «concepte» també hi estem incloent possibles pàgines web que són text i no executables pròpiament dits.

2.3.1.3. Registre d'activitats

Amb vista a tenir una visió general del que passa en el nostre sistema i en els continguts que servim, hem de complementar el control d'accessos tractat anteriorment amb un registre complet de les diferents activitats o accions que fan els usuaris sobre el nostre servei. Tenim dos grups diferenciats sobre els quals hem de fer el registre i que són independents, tant en interès com en forma de controlar-los:

- Activitats sobre el sistema: aquestes activitats són les que afecten directament el nostre sistema, servidors i components que ens ajuden a oferir el nostre servei. Hem de desar un registre de totes les accions que realitzi un usuari des del moment en què es presenta o autentifica en el nostre sistema fins al moment en què l'abandona. És important que quedin registrats totes les ordres i accions que realitza, i també la data i l'hora que els realitza. Amb aquesta informació podrem buscar qualsevol cosa estranya que passi en el nostre sistema i localitzar el responsable. A manera d'exemple, podríem localitzar l'usuari que ha esborrat un fitxer determinat o l'últim usuari que ha accedit a un fitxer de configuració d'un dels nostres servidors i si l'ha modificat. Com més gran sigui el grup de treball que s'encarrega del manteniment del nostre servei web, més importància tindrà un registre detallat de tot el que s'hi realitza. I adquireix encara més importància si el manteniment l'ofereixen diferents empreses o departaments. Els mateixos sistemes operatius ens ofereixen eines per fer aquest tipus de registre, a més a més, en el mercat podrem trobar productes més potents que complementen o milloren els anteriors. En el cas dels servidors (web, aplicacions, bases de dades), la majoria de productes ens permeten activar el registre d'accions sobre aquests, de manera que, ja sigui en els *logs* o en fitxers especí-

fics, puguem veure totes les modificacions i accions que es realitzen i qui i quan les realitza.

- Activitats sobre els continguts: amb aquest grup volem comprendre les accions i modificacions que es realitzen en les aplicacions o continguts que servim. Aquestes accions afecten més els continguts en si que el sistema i, per tant, la forma o eina per fer-ne el registre ens l'ha de facilitar el mateix contingut. Com a subministradors del manteniment, per a nosaltres és molt més important el grup anterior, però pot ser que per al client sigui igual o més important saber qui ha fet què en les seves aplicacions. Normalment, aquest tipus de registre d'activitats és configurable i les aplicacions o els continguts se solen dipositar en fitxers específics o de *log* i, en alguns casos, es desen en la base de dades que ja usa el contingut esmentat.

Segons el volum d'accions i activitats que es realitzin sobre el nostre sistema, pot ser que el seu control generi una gran quantitat d'informació i, per tant, la forma de tractar-la i emmagatzemar-la ha de ser estudiada i establerta. Segons el registre que vulguem llegir, pot ser que el mateix fabricant del servidor o component que ha creat el registre ja ens faciliti una manera fàcil de tractar la informació, o pot ser que hàgim de buscar eines o aplicacions en el mercat per tractar aquestes dades. Ara mateix no hi ha cap eina que serveixi com a propòsit general per tractar aquest tipus d'informació, ja que en la majoria dels casos es tracta de fitxers de text pla i, per tant, els podem llegir amb infinitat d'aplicacions, encara que ens pugui interessar alguna en concret que ens permeti formatar-los al nostre gust perquè així s'adaptin més a les nostres necessitats.

2.3.1.4. Publicació de continguts

Com a responsables del manteniment del servei web, pot caure sota la nostra responsabilitat la publicació i actualització dels continguts que servim. Aquest procés, en línies generals, consistirà en la instal·lació, actualització o eliminació d'un contingut o una aplicació que estiguem servint des del nostre sistema. Els responsables de la creació d'aquests continguts són els que ens han de facilitar el paquet que contingui els arxius que hem de servir. A més a més,

és recomanable i fins i tot obligatori que juntament amb aquests paquets vingui una guia d'instal·lació o procediment de canvi, en què se'ns indiquin els passos que hem de seguir per a la instal·lació i posada en marxa correctes del contingut.

En l'apartat de gestió de la configuració ja parlem de com serà d'important portar un control correcte de la versió dels continguts que tinguem en el nostre sistema; així doncs, podem fixar uns passos bàsics que cal seguir en qualsevol publicació que hàgim de realitzar i que, sense que es tracti d'una guia exhaustiva del procediment estàndard, s'adaptarà a la majoria de casos que ens trobem.

Aquests passos són els següents:

- Petició de publicació: en aquesta petició, qui la faci, ens ha d'indicar si es tracta d'un nou contingut, de l'actualització d'un de ja existent o de l'eliminació d'algun que estiguem servint actualment. És recomanable que també incorpori el motiu de la petició, ja sigui per correcció d'algun error en el contingut actual, millora d'aquest… També ha d'incloure el paquet amb els components que s'hi han d'instal·lar, i la versió del component esmentat, que ens ha d'ajudar a gestionar el nostre control de la versió i configuració. En aquest punt, també se'ns facilitarà la guia o el procediment d'instal·lació i configuració del contingut, i un pla de proves del seu funcionament correcte, una vegada realitzada la publicació. Un altre punt important que s'ha de tenir en compte en la petició és si el contingut que s'ha de modificar afecta directament o indirectament altres continguts i si, per tant, aquests també es veuran afectats pel canvi.
- Planificació de publicació: atès que els continguts que estem servint són utilitzats per usuaris i la seva actualització molt probablement comporti una parada o pèrdua de servei d'aquest, hem de negociar amb el client o amb el departament adequat en quin moment podem fer la publicació. Recordem que haurem de complir un acord de nivell de servei i que, per tant, una pèrdua de servei d'una aplicació sense una planificació prèvia d'aquesta ens podria portar a no complir l'acord. Amb la informació que ens hagin passat en la petició anterior i amb la nostra experiència, haurem de fixar quant de temps necessitarem per elaborar la publicació i així informar els interessats de manera que puguin decidir quan els va millor tenir la indisponibilitat.

- Execució del treball: Segons de si es tracta d'una nova instal·lació, una actualització o una eliminació de continguts, tindrem diferents subfases. El procés més llarg seria el d'actualització o modificació i, per tant, descriurem aquestes subfases i, posteriorment, detallarem les que ens hem de saltar segons el procés:
  - –Parada del contingut: hem de fer els passos necessaris perquè el contingut deixi d'estar disponible als usuaris, així com altres continguts que es puguin veure afectats. D'aquesta manera ens assegurem que, en cas que es desin dades, aquestes no es modifiquin durant les fases següents.
  - –Còpia de seguretat: abans de modificar res, hem de fer una còpia de seguretat de tot el que s'afectarà amb la instal·lació (principalment dades de configuració i d'usuari). Aquesta còpia la desarem sabent la versió corresponent del contingut per poder tornar-hi en cas que calgui.
  - –Instal·lació, eliminació o modificació del contingut: en aquest punt haurem de fer la publicació del nou contingut. A aquest efecte, usarem el paquet d'instal·lació que ens hagin facilitat i la guia.
  - –Arrencada del contingut: després de la instal·lació arrencarem el contingut. És recomanable que el nou contingut no sigui públic als usuaris fins que no fem les proves corresponents. Per a això, el nostre servidor web o servidor d'aplicacions ens haurà d'oferir la possibilitat de publicació per a un grup específic d'usuaris o per a una xarxa específica. No hem d'oblidar arrencar altres possibles continguts afectats que hàgim aturat prèviament.
  - –Validació del contingut: seguint la guia que se'ns hagi facilitat o els nostres coneixements sobre el funcionament del contingut, haurem de provar i validar el seu funcionament correcte. Segons si les proves surten correctament, farem públic el nou contingut i finalitzarem així el procés.
  - –Marxa enrere: en cas que les proves no hagin anat bé o ens trobem algun comportament anòmal, tant nostre com dels usuaris, hem de valorar amb la resta d'interessats la possibilitat de tornar a la versió anterior. Aquest procés s'ha de fer després tot seguit dels anteriors, en cas que nosaltres trobem fallades segons la nostra bateria de proves, però també es pot fer a posteriori si són els usuaris els qui troben les fallades. Si ens trobem en aquest últim cas, el procés serà més complex, ja que haurem de tornar a aturar el contingut, instal·lar la versió anterior…

Si es tracta d'una nova instal·lació, els dos primers passos no caldran i la marxa enrere comporta l'eliminació del nou contingut o treure la visibilitat pública cap als usuaris. Si, per contra, eliminem un contingut, no hem de fer una validació i la marxa enrere serà un procés d'instal·lació nou d'aquest.

2.3.1.5. Neteja i rotació de *logs*

Com ja hem apuntat en l'apartat de còpies de seguretat, els *logs* que desem de tots els elements del nostre servei poden representar una gran quantitat d'informació i, per tant, hem de definir una política correcta per tractar-los i així no saturar el nostre sistema d'emmagatzematge i servidors d'informació innecessària. Com la resta d'apartats, no hi ha una norma clara per crear una política correcta, ja que dependrà molt de cada situació particular, però, com sempre, intentarem donar unes referències generals que ens podran ajudar a prendre una decisió correcta. De cada *log* que volem desar hem d'analitzar els punts següents:

- Quantitat d'informació per dia que s'hi desa (podem calcular-ne una mitjana analitzant diversos dies).
- Quants dies necessitem tenir accessibles des del nostre servidor de producció abans de procedir a l'arxivament.
- Quants dies d'històric del *log* ens interessa tenir desats.

Si tenim la informació anterior, podrem definir una política de rotació i neteja correcta. Segons la quantitat d'informació que es generi en el *log* i de quants dies vulguem desar-la en el seu servidor, coneixerem l'espai que necessitarem en el servidor o sistema d'emmagatzematge on ho anem a desar. Amb els dies d'històric que vulguem desar en el nostre sistema de còpia de seguretat, sabrem l'espai que hi necessitarem per tenir aquesta informació.

Normalment, els dies diferents del que està en curs que tinguem desats en producció estaran en fitxers comprimits per tal d'estalviar-se el màxim d'espai possible. De la mateixa manera, aquests fitxers comprimits seran els que vagin a parar a les nostres còpies de seguretat.

També hem de definir una normalització per als noms dels fitxers de *logs* que anem desant. Normalment en el nom estarà reflectida la data a la qual correspon el *log*, i fins i tot l'hora en cas de tenir una rotació superior a la diària.

A tall d'exemple per entendre com funcionaria una gestió de *logs,* confeccionarem una per a un servidor web genèric, el qual hem configurat perquè desi el log en la ruta /var/logs/http/server1.log (suposem que el servidor web està arrencat sobre un Linux) i hem calculat que, de mitjana, el fitxer ocupa 30 Mb al dia. Com a decisió de disseny, volem desar en el servidor els *logs* de l'última setmana i, en la nostra còpia de seguretat, els últims tres mesos, per la qual cosa, en cas de no voler comprimir els fitxers, hem de reservar en el nostre sistema de fitxer 8 x 30 Mb (els set anteriors més el dia en curs) i en el nostre sistema d'emmagatzematge de còpies 90 x 30 Mb (no seria lògic desar una còpia dels fitxers sense comprimir; per tant, haurem de veure la mitjana de ràtio de compressió dels fitxers i així calcular un espai més d'acord). A més a més, hem de decidir una nomenclatura per als fitxers que ens ajudi a saber de quin dia són. Hi posarem la següent server1_dd_mm_yyyy.log, en què *dd* és el dia del mes, *mm* el mes i *yyyy* l'any. Per fer la rotació dels xxxxx?? hem de tenir un *script* o programa que copiï el fitxer del dia en curs a un nou fitxer amb la data, buidi el fitxer del dia en curs i copiï el fitxer de fa més de set dies al nostre sistema de *backup* i després l'esborri del servidor. Amb aquest sistema un 04 de juliol de 2011 tindrem els fitxers següents en el nostre servidor:

/var/logs/http/server1_27_06_2011.log
/var/logs/http/server1_28_06_2011.log
/var/logs/http/server1_29_06_2011.log
/var/logs/http/server1_30_06_2011.log
/var/logs/http/server1_01_07_2011.log
/var/logs/http/server1_02_07_2011.log
/var/logs/http/server1_03_07_2011.log
/var/logs/http/server1.log

I el fitxer més antic que podrem recuperar de la nostra còpia de seguretat serà:

/var/logs/http/server1_04_04_2011.log

2.3.2. La supervisió del servei web

Una part fonamental del manteniment d'un servei web és supervisar-lo. La supervisió ens valdrà per conèixer en tot moment l'estat del nostre servei, avi-

sar-nos en cas que es produeixi algun error o prevenir-nos abans que aquest passi. Amb un bon sistema de supervisió, podrem minimitzar els errors del nostre servei i aquesta és la millor manera de garantir una alta disponibilitat d'aquest.

2.3.2.1. Elements que s'han de supervisar

Dins d'un servei web hi ha nombrosos elements que hem de tenir en compte per garantir una bona qualitat de servei. És important localitzar aquests elements i fer-ne una supervisió correcta i efectiva. Segons la mida del nostre sistema o la quantitat de components que tingui el nostre servei, n'haurem de supervisar uns o altres elements; al seu torn, és important tenir clar el que volem supervisar o el que és important per a nosaltres. Dividirem un servei web en els components fonamentals que el formen i així veurem quins elements hem de supervisar:

- Sistema: és la base del nostre servei i, per tant, una de les parts més important que cal tenir present a l'hora d'aconseguir una supervisió efectiva. Dins d'aquest apartat, ens trobem els elements fonamentals següents:
  - –Estat del servidor físic: hem de monitorar si el nostre servidor està encès i funcionant correctament.
  - –*Logs* del sistema operatiu: tots els sistemes operatius ens ofereixen una sèrie de *logs* on podem veure tot el que hi passa. La informació que s'escriu en aquests fitxers sol ser configurable i hem d'activar tot el que ens pugui interessar supervisar. Entre altres coses, en aquests *logs* trobarem possibles errors de maquinari del servidor físic, errors del sistema de fitxers, errors concrets del sistema operatiu… En el cas dels servidors UNIX i Linux els *logs* de sistema els podem trobar en /var/adm/syslog/, /var/adm i /var/log. Aquests fitxers contenen paraules clau com *info*, *warning* o *error*, que ens indiquen informació relativa al nostre sistema i al nostre maquinari, i ens mostren la gravetat del missatge segons la paraula que continguin. En el cas de sistemes Windows, aquesta informació la podem trobar en les eines administratives del sistema, on també trobem el gestor d'esdeveniments.
  - –Processos del sistema: en un servidor hi haurà molts processos corrent. Ens interessa que alguns d'aquests processos, fonamentals per al nostre servei, estiguin supervisats. D'aquesta manera, si algun deixa de funcionar ens

n'assabentarem per poder corregir el problema en el menor temps possible. Per veure els processos que estan corrent en un sistema Unix o Linux, disposem de l'ordre «top», que ens ensenya els processos esmentats i la quantitat de memòria i CPU que estan consumint en aquell moment. En el cas de Windows, podem trobar aquesta informació a l'administrador de tasques.

–Ocupació del sistema de fitxers: tots els sistemes operatius tenen un sistema de fitxers on es desa la informació tant del sistema com de les aplicacions i els programes. És important tenir controlat el nivell d'ocupació del sistema esmentat i fixar alarmes que ens avisin quan una partició o unitat concreta és pròxima al seu ple o nivell del 100% d'ocupació. No oblidem que hi ha particions fonamentals per al sistema i, si arriben a omplir-se, podrien provocar una caiguda total d'aquest. Per veure la informació d'ocupació del sistema de fitxers en sistemes Unix o Linux, disposem de l'ordre «df». En el cas de Windows, podem veure l'ocupació de les unitats des d'«El meu ordinador» o «El meu equip», segons la versió.

- Servidor web: com ja hem comentat diverses vegades, tot sistema de serveis web tindrà un servidor web i segons la quantitat d'instàncies que tinguem d'aquest, és fonamental supervisar-lo, ja que ens podria portar a una pèrdua de la disponibilitat total del nostre servei. Els elements fonamentals que s'han de supervisar en aquests servidors són:

  –*Logs*: tots els servidors web escriuen fitxers de *logs* en què desen informació important per ser supervisada. Hem de monitorar aquests fitxers en els quals, entre altres coses, podrem trobar errors o excepcions de la instància web, intents d'atacs contra el servidor, pàgines no trobades sol·licitades al servidor, caigudes d'instàncies… Com que Apache és el més conegut, ens hi podem centrar per analitzar els seus *logs*. Per saber com es diuen els nostres fitxers de *logs* en la instal·lació que tenim d'Apache, ho podem veure en el fitxer de configuració httpd.conf. Per defecte aquests fitxers s'anomenaran access_log i error_log. Dins d'aquests fitxers trobarem tota la informació relativa a accessos, peticions i errors que generi la nostra instància web. A tall d'exemple, si trobem molts *get* seguits sobre el mateix recurs en un període de temps molt petit (un segon), podrem estar patint un atac per allau de peticions sobre el nostre servidor. Si, per contra, veiem molts missatges 404 Not Found, és possible que algun dels continguts que servim estigui encaminant els usuaris a un recurs que no existeix i que, per tant, tinguem un error en el dit contingut. Hem de

prestar especial atenció a les línies del *log* que continguin les paraules *error* o *warning*.

–Processos: hem de conèixer els processos que componen el nostre servidor i quins han d'estar arrencats en el nostre sistema perquè aquest proporcioni un servei adequat. Pot ser que ens trobem processos que s'anomenin igual, però que n'hàgim de tenir un nombre determinat arrencats o un mínim d'aquests.

–Servei: a més a més de tot això i per garantir que una instància concreta està donant un servei correcte, se solen posar pàgines de test perquè puguem supervisar que la instància web està servint correctament aquesta pàgina cada cert temps. En cas que no puguem accedir al contingut bàsic esmentat, significarà que hi ha un problema.

- Servidor d'aplicacions: en cas que el nostre servei disposi d'aquest element, hem de procedir igual que en el cas del servidor web i supervisar tant els *logs* d'aquest com els processos que el componen. En aquest cas, el contingut mínim que se sol comprovar és una petita aplicació que ens torni un missatge concret; per tant, comprovarem que el nostre servidor està treballant correctament.
- Base de dades: si el nostre sistema disposa d'una base de dades, la pèrdua de servei o errors en aquesta poden provocar que els nostres continguts deixin de funcionar; així doncs, com en els casos anteriors també hem de supervisar correctament el component esmentat. Les parts fonamentals que s'han de supervisar són:

  –*Logs*: com en els casos anteriors, els servidors de bases de dades tenen diversos fitxers de *logs* en què podrem trobar informació valuosa sobre la seva activitat, com ara errors en l'arrencada o parada, errors en l'accés a les dades, problemes de memòria o rendiment… Atès que el mercat està dominat avui dia per Oracle, podem analitzar el log d'aquest servidor a tall d'exemple. Aquest *log* s'anomena alert.log i és dins les carpetes en la qual estigui instal·lat el servidor (és configurable; per tant pot variar d'una instal·lació a una altra). Aquest *log* conté els errors que generi la nostra instància, aquestes línies contenen la paraula *error* i el tipus d'error que comença per ORA– seguit d'un número identificatiu. També hi podrem trobar informació sobre l'arrencada, la parada i el funcionament de la nostra instància. Com la resta de casos, cal prestar especial atenció a les línies que continguin les paraules *error* o *warning*.

–Processos: haurem de conèixer els processos concrets i el nombre d'aquests que componen el nostre servidor i supervisar-los, de la mateixa manera que en la resta de casos.
–Ocupació: tots els servidors de bases de dades tenen les seves pròpies estructures de dades per emmagatzemar la informació. És important que controlem el nivell d'ocupació d'aquesta, de la mateixa manera que ens passava en el cas del sistema de fitxers.
–Servei: en aquest cas, la manera de comprovar que estem servint correctament les dades se sol fer mitjançant una connexió periòdica a la instància de base de dades que vulguem supervisar.
–Continguts: a més a més dels components del nostre servei, ens pot interessar supervisar continguts o aplicacions concretes. Per fer aquesta supervisió el fabricant o distribuïdor del contingut ens haurà de facilitar la informació necessària per dur a terme aquesta tasca. Detalls com ara si el contingut desa *logs* i on.

2.3.2.2. Mètodes de supervisió

Quan parlem de mètodes de supervisió ens referim a la manera com hem d'actuar sobre el nostre sistema per corregir errors o evitar que aquests s'arribin a produir. Podem dividir en dos grups generals els mètodes de supervisió actuals:

- Reactiu: aquests són els més usats actualment i consisteixen en el monitoratge del nostre sistema, controlant quan s'hi produeix una fallada o error. Quan això passa es produirà una alarma en el nostre sistema o aplicació de supervisió que ens avisarà de l'esdeveniment i, després d'això, podrem actuar sobre el sistema per solucionar la situació i que l'error desaparegui. El monitoratge es realitza mitjançant la lectura de *logs* i el control de processos, tal com hem explicat en l'apartat anterior, buscant situacions d'error o risc dins d'aquests.
- Proactiu/predictiu: aquests sistemes són més complexos i requereixen una quantitat superior de dades que els anteriors. En aquest cas, no solament supervisarem els possibles errors que apareguin en els nostres *logs* o sistema, sinó que també desarem tota la informació de què disposem en el sistema en el moment que es produeix l'error i abans d'aquest. Amb tota aquesta informació, el nostre sistema ens podrà avisar de quan hi ha la possibilitat que es produeixi un error, de manera que puguem actuar abans que aquest

s'esdevingui i evitar així la indisponibilitat. Veient això, sembla que aquests sistemes són els ideals i els que haurien de predominar en el mercat, però no hem d'oblidar que no sempre es pot prevenir un error i que, sovint, la quantitat de variables que necessitem per portar a producció aquests sistemes és molt difícil de determinar, o fins i tot poden ser massa.

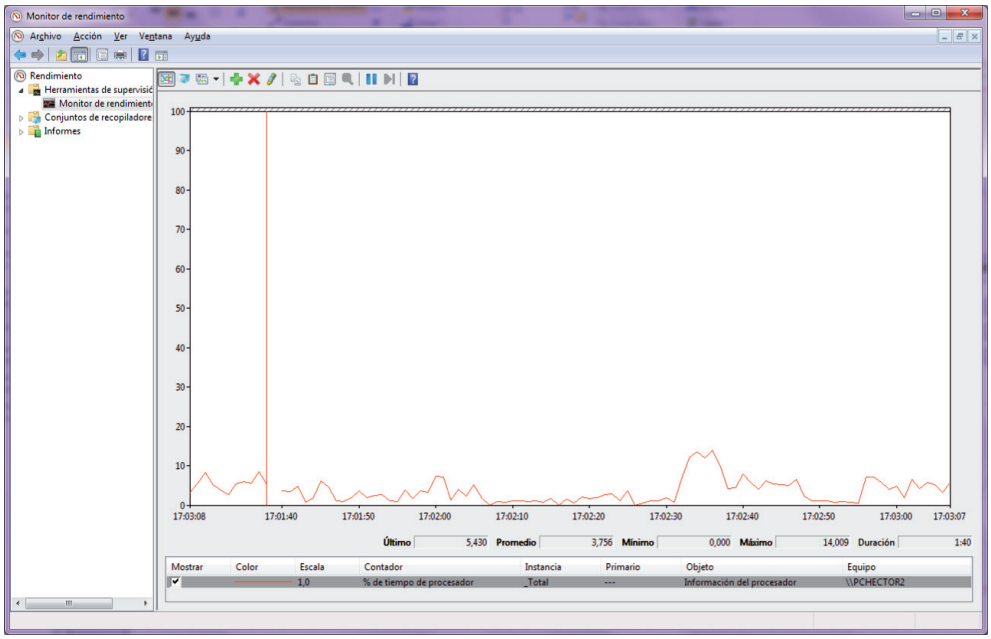
Com sempre, hem d'analitzar el cas particular en què ens trobem i adoptar la solució que millor s'adapti al nostre sistema. Sempre serà millor poder evitar que es produeixi un error a haver de corregir-lo quan es produeixi; per tant, podem optar per un sistema mixt que ens pugui prevenir dels errors fàcils de predir i avisar dels que ja s'han produït com més aviat millor. Aquesta última opció és la que s'implementa en la majoria de sistemes i la que incorporen les principals eines tants comercials com gratuïtes.

2.3.2.3. Gestió de la capacitat

Per assegurar-nos que el nostre sistema està funcionant correctament i que, per tant, estem oferint un bon servei, no n'hi ha prou solament amb supervisar els elements susceptibles d'error, sinó que també hem d'analitzar i conèixer en quin nivell de capacitat es troben els recursos que componen el nostre sistema.

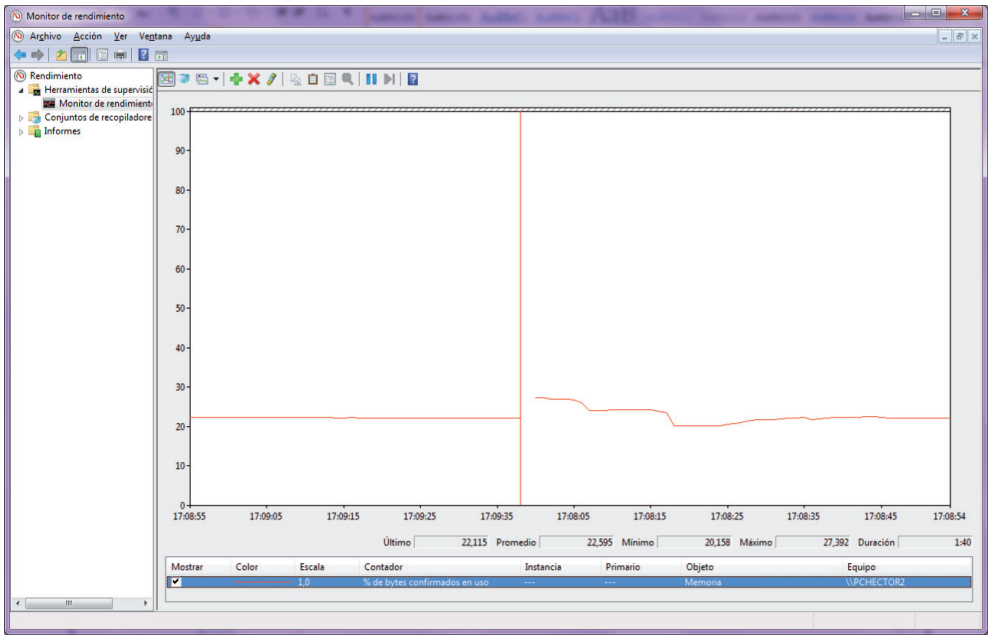
La gestió de la capacitat consisteix en la supervisió, l'anàlisi i l'estudi del rendiment i consum de recursos que tenen els elements que componen el nostre sistema per aplicar-hi correccions o millores. Els recursos següents són els que haurem de tenir presents en qualsevol sistema:

- CPU: tant els processos del nostre sistema operatiu com els dels components del nostre servei web (servidor web, servidor d'aplicacions…) consumiran temps de processador del servidor on s'estiguin executant. Si el consum dels nostres processadors es troba al 100%, significarà que els processos no s'estan executant amb la fluïdesa en què ho haurien de fer i que, per tant, estem tenint retards i un funcionament dolent en les nostres aplicacions. Es considera que, quan un servidor es troba a un 80% de consum sostingut de CPU, ja està saturat i cal pensar en una possible ampliació o canvi. El canvi o ampliació s'ha de fer quan s'hagi verificat que no sigui el mal funcionament d'un o diversos processos el que està provocant l'alt consum de processador i que, per tant, corregint aquests processos, acabem amb el problema.



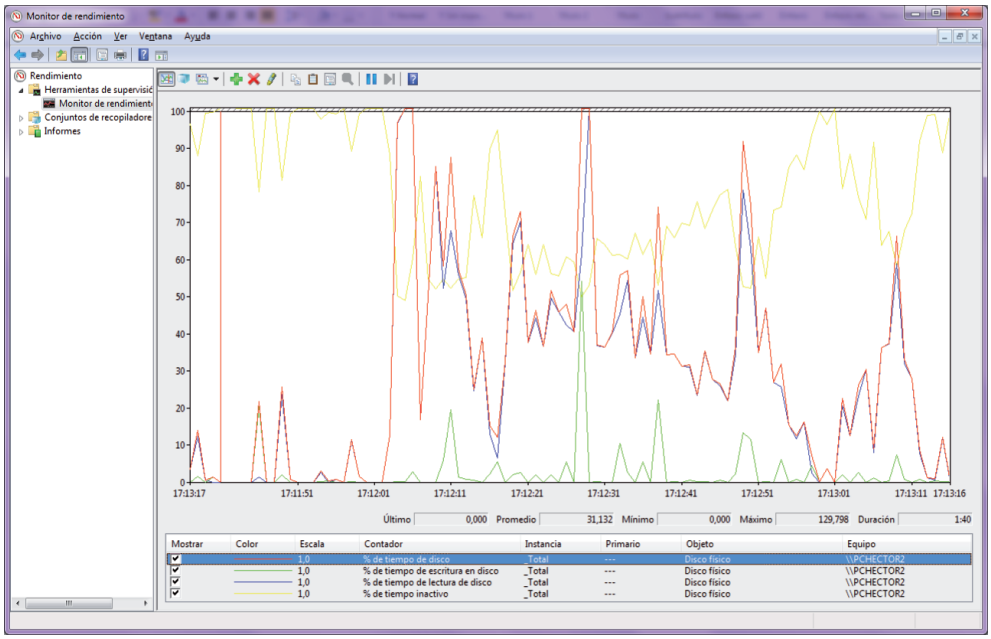
En la imatge, una gràfica del consum de CPU d'un servidor sense saturació treta usant les eines de Windows.

- Memòria: tots els processos que s'estan executant en un servidor i gran part de la informació a la qual accedeixen estan desats en la seva memòria. Si arribem al màxim d'aquesta, podem provocar fallades en els nostres processos, de manera que aquests es deixin d'executar i provocar, fins i tot, un col·lapse total del sistema. Per aquest motiu és important supervisar el percentatge de memòria que estem usant. Es considera que quan un servidor està consumint el 100% de la seva memòria física i ha de començar a usar la memòria virtual es troba saturat i cal ampliar la memòria o canviar completament el servidor. Abans d'arribar a aquesta mesura, la majoria de vegades és recomanable analitzar quins processos són els que estan consumint més memòria i si aquest comportament és correcte o no per a aquests.



En la imatge, una gràfica del tant per cent de consum de memòria d'un servidor sense saturació treta usant les eines de Windows

- Temps d'entrada/sortida a discos: tant si tenim discos interns en els nostres servidors com si són sistemes d'emmagatzematge externs connectats a aquests, hem de tenir sempre present que molts dels nostres processos faran operacions de lectura i escriptura sobre aquests. Hem de controlar els temps que tarden aquestes operacions, ja que poden ser un coll d'ampolla important per a l'execució correcta dels nostres components. Aquestes mesures són especialment importants en servidors que tinguin un servidor de bases de dades o processos amb un ús intensiu de discos.
- Amplada de banda: atès que l'objectiu d'un servei web és proporcionar continguts a usuaris remots a través d'una xarxa, és molt important per a aquest controlar el consum d'amplada de banda de la xarxa a la qual estigui connectat. Si observem que estem consumint un 100% de la capacitat contractada o limitada per la xarxa, hem de prendre mesures urgentment, ja que el nostre servei no ha de ser mai el que limiti la velocitat a la qual els usuaris poden accedir als continguts. Les solucions poden passar per contractar més amplada de banda, canviar la tecnologia de xarxa que estem usant…



En la imatge, una gràfica del tant per cent del temps d'accés a discos d'un servidor sense saturació treta usant les eines de Windows

> No hem d'oblidar, abans de prendre aquestes mesures, que de vegades un mal ús per part d'un usuari o un atac sobre el nostre sistema pot arribar a consumir tota la nostra amplada de banda sense que necessàriament sigui un problema de capacitat.

La majoria dels sistemes operatius del mercat ens ofereixen eines per poder recollir dades sobre els consums comentats anteriorment i poder-ne fer l'anàlisi oportuna. També en el mercat, trobem eines tant comercials com gratuïtes per recollir aquestes dades o per fer-ne una anàlisi més profunda.

2.3.2.4. Estadístiques de rendiment del servei

Seguint en la línia de l'apartat anterior, a més a més de la prevenció i correcció d'errors, hem de tenir en compte la qualitat de servei que està oferint el nostre servei web. Anteriorment, hem vist que per a això hem de monitorar i controlar els recursos de què disposem. Ara ens centrarem en el rendiment final que ofereix el nostre servei i el que realment observen els usuaris d'aquest.

En l'apartat d'estadístiques d'accés als continguts, ja tractem aquest tema en profunditat i, per tant, ara només hem d'afegir que recollir les estadístiques de rendiment del sistema per a monitorar-les i buscar possibles millores, en cas que el comportament no sigui el desitjat, és una tasca que correspon a la supervisió del sistema.

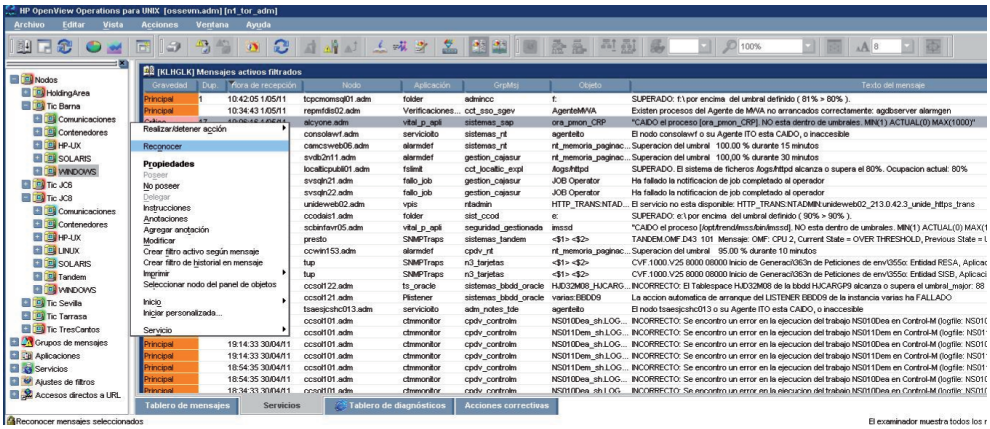
Quan es realitza un estudi de rendiment sobre un servei web se solen estudiar tant les estadístiques d'accés com les de capacitat del sistema per treure una conclusió de com funciona aquest i aportar-hi solucions per millorar-lo.

2.3.3. Eines de diagnòstic

Per fer tots els treballs que requereix el manteniment d'un servei web i el sistema on està allotjat, utilitzarem una sèrie d'eines per ajudar-nos a dur-ho a terme d'una manera correcta. Intentarem posar alguns exemples de les eines que trobem actualment al mercat per exercir les principals tasques de què hem parlat anteriorment:

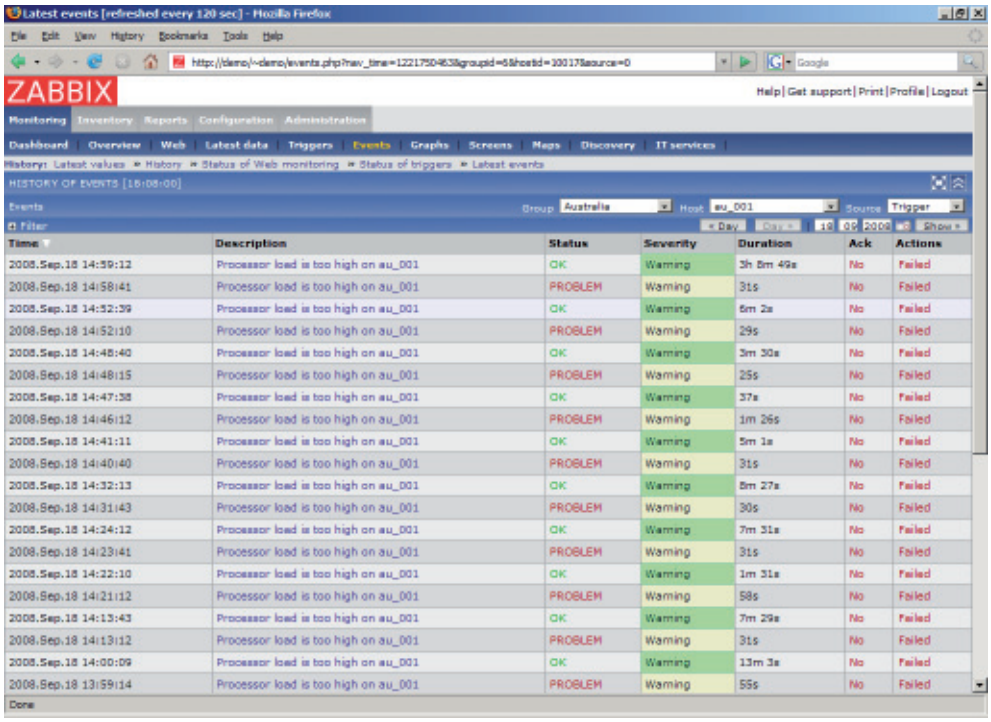
- Supervisió del sistema: per fer totes les tasques descrites que necessitarem per supervisar el nostre sistema, com ara lectura de *logs*, control de processos, ocupació d'espai en sistemes de fitxers... en el mercat podem trobar diverses eines.
  - La més usada a nivell comercial en la majoria de sistemes (fins i tot encara que no siguin per a serveis web) és HP Openview Operations, més conegut com a OVO. Aquesta eina ens permet recol·lectar la informació que li indiquem dels nostres servidors i enviar-la a un sistema centralitzat on es desarà; a més a més, ens proporciona aplicacions per veure a cada moment el que estem monitorant i les alarmes o esdeveniments que tenim actius en tots els nostres servidors, i per configurar nous monitoratges o esdeveniments. Empreses importants solen utilitzar altres eines per a gestió d'incidències sobre els seus sistemes de tecnologies de la informació, ja que OVO ens proporciona interfícies de comunicació amb les principals eines perquè, d'una manera automàtica, un esdeveniment o alarma creï una incidència en les eines externes esmentades. A més a més de tot això, OVO és compatible amb la majoria de sistemes operatius, la qual cosa ens pot limitar molt amb altres eines. Podem resumir que aquesta eina ara mateix controla el mercat comercial i que pot ser una opció segu-

ra per a sistemes importants i grans, encara que no hem d'oblidar els costos de llicències d'aquest producte, que poden encarir notòriament la nostra solució.



En la imatge, la consola OVO mostrant els missatges actius de grup de servidors

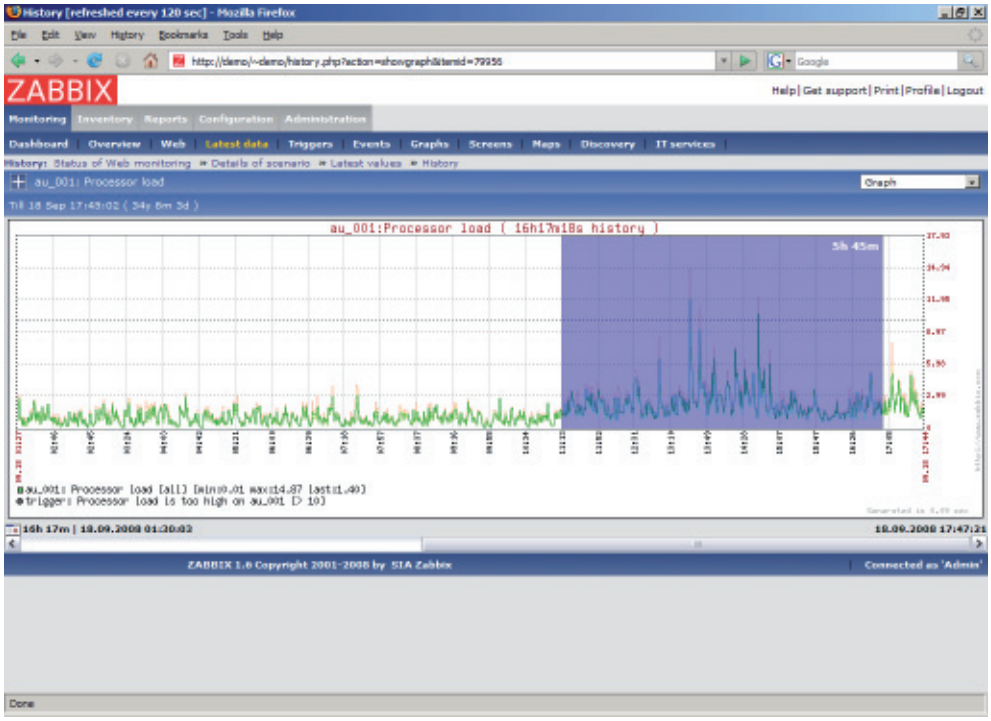
  - D'adquisició gratuïta i de codi obert, en el mercat podem trobar diverses solucions, encara que cap s'imposa de manera tan aclaparadora per damunt de la resta com en el cas comercial. A tall d'exemple, hi destacarem la solució Zabbix, que implementa la majoria de les característiques que ens ofereix la solució comercial tractada i també és compatible amb gairebé tots els sistemes operatius. La forma de funcionament és similar al producte anterior, amb agents o programes instal·lats en els servidors que volem monitorar i un sistema centralitzat col·lector d'informació al qual ens connectarem amb les eines que ens proporcionen per obtenir la informació i veure la situació del sistema. La instal·lació i posada en marxa d'aquest producte pot resultar una mica més complicada que la solució comercial, però hem de tenir-la present molt especialment si el nostre projecte és petit o té un pressupost reduït.
- Gestió de capacitat: en aquest cas, ens interessa la recol·lecció periòdica de les dades de capacitat dels nostres sistemes i poder tractar aquestes dades d'una manera còmoda i intuïtiva. Moltes vegades és més fàcil analitzar una gràfica que una gran quantitat de dades en un fitxer. És, doncs, una de les característiques que demanarem a qualsevol solució que usem per a aquesta tasca.



En la imatge, la consola OVO mostrant els missatges actius de grup de Zabbix

– Des del punt de vista comercial, ens tornem a trobar amb HP OpenView i el seu complement MeasureWare. Aquesta solució s'encarrega, mitjançant uns agents instal·lats en els servidors i usant les eines que ens proporciona el sistema operatiu en què resideixin, de recollir totes les dades que li indiquem per a la seva anàlisi posterior. Amb les dades, podem treure gràfiques de pràcticament qualsevol cosa que afecti el nostre sistema en el període de temps que li indiquem, fins i tot ens permet en una mateixa gràfica comparar el comportament de diferents servidors respecte d'un recurs determinat, o posar diversos recursos d'un servidor i així tenir-ne una visió completa en una única imatge. Aquesta solució també ens proporciona un entorn web per poder fer les consultes sense necessitat d'aplicacions instal·lades en el nostre lloc de treball.
– Pel que fa a codi obert, ens trobem la mateixa situació que en l'apartat anterior. Hi ha diversos productes interessants, però cap predomina actualment en el mercat. La solució Zabbix també ens servirà per al que neces-

sitem en aquest apartat, ja que implementa gràfiques i la recol·lecció de les dades necessàries per a una bona gestió de la capacitat. Podem esmentar altres solucions, com Munin, que encara que una mica menys potent que l'anterior i més limitada pel que fa als sistemes operatius en què es pot instal·lar, també ens pot ser útil per a l'objectiu que busquem en aquest apartat, per l'adaptabilitat que presenten les seves gràfiques de recursos.



En la imatge, mesurament de càrrega de CPU d'un servidor usant Zabbix

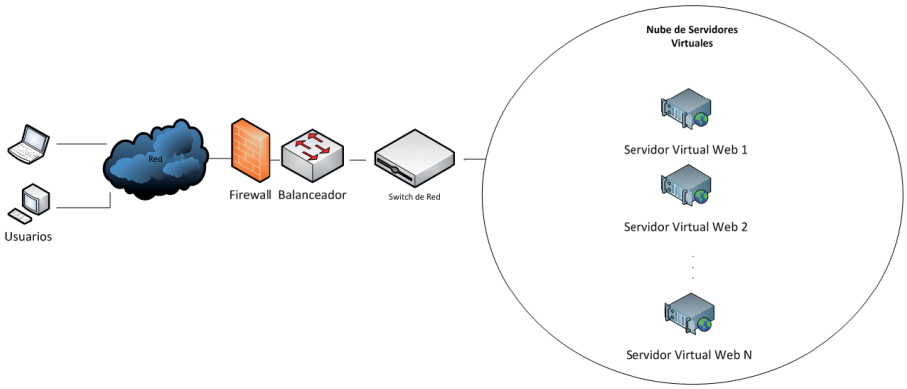
- Estadístiques de rendiment: la millor manera de fer aquests mesuraments és tenir agents externs que facin una varietat de proves sobre els nostres continguts per analitzar-ne els temps de resposta i el rendiment. Tal com esmentem anteriorment, la solució que s'imposa en aquests moments en el mercat és Google Analytics. Mitjançant la inclusió d'unes línies de codi a les nostres solucions i gràcies a les eines que ens proporciona, podrem veure no solament proves puntuals, sinó estadístiques clares de com accedeixen els

usuaris als nostres continguts, quins són els més vistos i quins pateixen més retards. A més a més de la solució comercial, en molts casos i segons els continguts que servim, serà molt recomanable crear petits programes que accedeixin d'una manera més precisa a les parts que volem de les nostres aplicacions i mesurin els temps de resposta d'aquestes.

Recordeu que el mercat evoluciona molt ràpid i el que hem intentat en aquest apartat és donar una petita pinzellada a la situació actual i a algunes de les solucions que ens ofereix. Com sempre, el millor és consultar i buscar en el moment que ho necessitem la solució que més s'adapti al projecte que ens hagin encarregat i que hem d'implementar.

2.4. Producció automatitzada d'un servei web

Fins a aquest punt, hem vist què és un servei web, com dissenyar-lo i posar-lo en producció i, finalment, com mantenir-lo i fer-hi una millora constant. En aquest apartat intentarem explicar com minimitzar la intervenció humana sobre el servei automatitzant el màxim nombre de tasques possibles. Hem de veure l'automatització d'un servei com la situació ideal en què aquest servei és capaç d'adaptar-se als canvis, autocorregir-ne el comportament i solucionar d'una manera autònoma els problemes o errors que hi puguin sorgir. Per a qualsevol empresa, és important l'estalvi de costos; per tant, tenir una solució en què no hi hagin d'intervenir persones, amb l'estalvi que això representa, serà més que benvinguda, però no hem d'oblidar que arribar a una automatització total, en molts casos, sol ser una cosa utòpica i en algun moment sempre farà falta la supervisió o correcció de comportaments per part de persones. Sigui com sigui, el nostre objectiu en anar cap a l'automatització és que aquest tipus d'intervencions siguin mínimes, de manera que minimitzarem les persones dedicades a aquesta tasca. També hem de tenir present que la inversió inicial per construir un sistema autònom serà molt superior a la d'un sistema tradicional, sobretot si dins de l'empresa no disposem ja de les eines necessàries a aquest efecte i, per tant, cal valorar si a la llarga ens compensa la inversió.



En la imatge, un sistema muntat sobre una núvol de servidors virtuals que creix/decreix segons les necessitats del servei

Descriurem tot seguit els punts més importants que s'han de tenir en compte a l'hora d'automatitzar un servei web:

- Reacció a errors: és fonamental que un sistema automatitzat reaccioni davant possibles fallades dels elements que el formen i les puguin corregir d'una manera autònoma. Per aconseguir aquest comportament, hem de disposar de productes que siguin capaços de fer-ho pel seu compte o desenvolupar-los nosaltres. En cas d'optar per l'opció de crear-los nosaltres mateixos, hem de tenir presents tots els possibles errors que es poden produir en el nostre sistema i definir i implementar la solució que s'executarà automàticament quan això passi.
- Reacció a càrrega de treball: un altre aspecte que hem de valorar en un sistema d'aquest tipus és la possibilitat que el mateix sistema mesuri la càrrega de treball al llarg del temps i sigui capaç d'arrencar o parar nous recursos segons que la situació ho exigeixi. Podem desenvolupar un *script* que mesuri la quantitat d'usuaris que estan connectats a totes les nostres instàncies web i, segons el nombre d'aquests, aixequi o aturi noves instàncies. Els principals servidors d'aplicacions del mercat que implementen clústers i alta disponibilitat tenen utilitats que els permeten aixecar noves instàncies d'aquests i afegir-les al clúster que es trobi amb una càrrega de treball elevada que anteriorment hàgim prefixat.
- Reacció a falta de recursos: com hem parlat anteriorment, la falta de recursos pot fer que el nostre sistema no funcioni adequadament. Les solucions

que aportàvem eren l'ampliació del servidor afectat o trobar quin procés està causant el problema i solucionar-lo. En un sistema automatitzat, és important assegurar-nos que els processos que s'executen s'han provat suficientment i que no arribaran a produir aquest tipus de problemes. Per adoptar solucions automàtiques davant aquest problema, una de les opcions és muntar el nostre servei sobre un núvol de servidors virtuals, de manera que si ens quedem sense recursos, puguem, de manera automàtica, arrencar un o diversos nous servidors virtuals per repartir la càrrega de treball i acabar amb la falta de recursos.

- Reacció a problemes inesperats: hem de tenir sempre present que és molt difícil o impossible aconseguir un sistema autònom perfecte; per tant, hem de tenir controlada la situació d'error d'aquest. Si es produeix una situació o un error que el sistema no és capaç d'autocorregir, aquest ha de tenir definit un pla d'acció per alertar personal humà que pugui analitzar i corregir la situació. Dins d'aquest pla, i sempre que es pugui, hem de preveure la millora contínua del sistema, de manera que un problema que no estigui previst i que s'hagi de corregir manualment es pugui incloure dins de l'automatització per evitar que aquesta situació es produeixi en el futur.

Per veure com es modificaran les diferents fases del nostre projecte quan decidim que aquest ha d'estar automatitzat, analitzarem els canvis que farem en cadascuna o les decisions que haurem de prendre, a més a més de les ja estudiades en els capítols anteriors:

- Disseny: aquesta és una de les fases que més variaran si decidim que el nostre servei web estarà automatitzat. En totes les decisions que prenguem en aquesta fase (elecció de productes, infraestructura, alta disponibilitat…) hem d'afegir el nou factor de l'automatització. A tall d'exemple, quan ens toqui prendre la decisió de triar el servidor web, hi hem d'afegir un nou requisit per a aquest, que consistirà que el mateix servidor sigui capaç de detectar quan una de les seves instàncies cau o deixa de donar servei i la pugui aixecar automàticament. Si no trobem productes o complements per a aquests que facin aquestes accions, hem d'afegir als nostres costos, tant monetaris com de temps, el desenvolupament de complements o *scripts* que facin aquestes tasques d'automatització. Una part de l'automatització que ens pot resultar útil o necessària si el nostre entorn conté una solució d'alta disponibilitat és la facultat del mateix sistema per adaptar-se al nivell de

càrrega de treball, arrencant noves instàncies del servei que es trobi saturat o aturant les que no calguin, cosa que també haurem de tenir en compte a l'hora de triar productes per a la nostra solució o, com en el cas anterior, comptar amb això per valorar els desenvolupaments que haurem de fer realitzar. En definitiva, a totes les decisions explicades en el capítol de disseny, hi hem d'afegir el nou factor d'automatització i tot el que això comporta.

- Desplegament: en el procés d'instal·lació i desplegament, el major impacte que notarem a l'hora d'implementar un servei automatitzat és l'increment notori en els temps. Serà molt més costós pel que fa a hores instal·lar i configurar un servidor web automatitzat que un de tradicional. Com és lògic, haurem de configurar tots els nous elements que en el futur s'encarregaran d'autogestionar el servei per a cada un dels elements que componguin el nostre sistema. És important que aquest factor l'hàgim tingut en compte a l'hora de valorar i planificar el nostre projecte, ja que no solament el temps d'instal·lació serà més gran, sinó que també la quantitat de proves que caldrà fer abans de donar per bo el servei s'incrementarà notòriament. En aquesta fase també haurem d'instal·lar tots els nous components i *scripts* que hàgim desenvolupat o adquirit per a l'automatització i, per tant, serà un nou treball que haurem de fer i haurem d'haver tingut en compte.
- Manteniment: si hem fet correctament tot el treball anterior, aquesta fase hauria de ser automàtica i, per tant, els recursos que hi hem de dedicar seran mínims. Com hem esmentat anteriorment, cap sistema és perfecte i, per tant, no podrem prescindir de tots els recursos humans. La millor manera d'enfrontar-nos a aquesta fase a l' hora de poder predir costos i recursos serà disposar d'un nombre més alt de gent inicialment per afrontar la reacció davant problemes inesperats del sistema i polir-lo perquè cada vegada es donin menys situacions que el portin a reaccionar així. Al llarg del temps, el sistema estarà molt més polit i podrem anar prescindint dels recursos. Com que la fase que realment s'automatitza és aquesta, els productes que s'hi usen són els que més ens interessa que siguin capaços d'implementar sistemes d'automatització. Són els que hem esmentat en l'apartat anterior. Implementen de manera nativa la possibilitat d'executar ordres o accions quan es troben amb un esdeveniment concret, per la qual cosa si els configurem adequadament, farem que siguin capaços de reaccionar davant fallades sense intervenció humana. Aquesta tasca, en un principi, pot ser força complexa i tediosa, ja que, en molts casos, comportarà la implementació de *scripts*, tant per al monitoratge dels esdeveniments com per a les posteriors

accions que ha de prendre el sistema de monitoratge automatitzat que estem intentant configurar. Els sistemes predictius, com que tenen un nombre de variables més alt, es poden adaptar millor a sistemes automatitzats i tenir en compte un nombre de situacions més elevat que els reactius, però continua sent complicada la seva implementació, ja que com més variables tinguem en compte més situacions hem de controlar i implementar convenientment perquè el sistema pugui ser totalment autònom; per tot això, hem de valorar si ens compensa o no tenir un sistema d'aquest tipus o si aquest s'adapta a les nostres necessitats i al nostre pressupost.

Veient tot això, hem de reflexionar si realment convé des d'un principi que el projecte que hem de dur a terme sigui automatitzat. Moltes vegades és millor implementar un sistema tradicional amb vista a poder ser automatitzat en un futur; d'aquesta manera, reduïm el seu temps de posada en marxa i podem solucionar tots els problemes inicials que planteja qualsevol servei web abans d'endinsar-nos en la fase d'automatització, que té els seus propis problemes i riscos. Actualment les grans empreses intenten que els seus serveis web siguin com més automatitzats millor, però normalment parteixen de sistemes que ja posseeixen, que estan implementats i funcionant d'una manera tradicional amb un manteniment realitzat per personal i no pel mateix sistema.

Capítol III

Conceptes bàsics de seguretat informàtica

Marco Marín Martínez

3.1. Introducció

En l'actualitat, tots estem connectats. Els nostres ordinadors de casa, els nostres telèfons mòbils i, si tenim una empresa amb aplicacions web per a clients, també estan connectades a Internet, al món.

Aquesta connexió entre tots i tot ha provocat que algunes persones s'interessin per les nostres dades, altres per aprofitar-se dels nostres sistemes, altres simplement per utilitzar-los com a pont, és a dir, aprofitar per fer coses il·lícites... També ens poden deixar sense servei els nostres negocis, cosa que no és desitjable, ja que pot arribar a produir-nos pèrdues de diners i clients.

A causa d'això, cada vegada més s'està incidint en la necessitat de tenir departaments de seguretat informàtica dins els nostres departaments d'informàtica.

Aquests departaments o equips no solament han de tenir en compte la seguretat informàtica des del punt de vista de control d'accessos no desitjats als nostres sistemes d'informació. Aquests departaments també han de garantir la integritat i la disponibilitat de les infraestructures i les dades.

En els apartats següents veurem diferents enfocaments de la seguretat informàtica, la seguretat física i la disponibilitat de les infraestructures.

Justificació

Tenint en compte el que hem comentat anteriorment, sembla clar que la seguretat dels nostres sistemes i les nostres dades és important.

En els sistemes actuals es produeixen transaccions, s'emmagatzemen dades i qualsevol pèrdua o accés a aquestes per part de tercers no autoritzats pot provocar una pèrdua de diners, clients, escàndols públics, pèrdua d'imatge de la nostra empresa o organització, etc. Aquests motius ens porten a implementar mesures de seguretat en les nostres empreses.

Què entenem per seguretat?

Es pot entendre la seguretat com una característica d'un sistema que ens indica que aquest sistema està lliure de qualsevol perill, dany o risc, és a dir, que és infal·lible. Això, en els sistemes actuals, és difícil d'aconseguir. Es realitza una modificació de la definició de seguretat i passem a parlar de fiabilitat més que de seguretat. Per aquest motiu en alguns punts parlarem de sistemes fiables.

S'entén que un sistema segur consisteix en un sistema que garanteix les característiques següents: confidencialitat, integritat i disponibilitat.

- Confidencialitat: ens diu que als sistemes i les seves dades només poden accedir persones o elements autoritzats i que les persones o elements autoritzats no utilitzaran de manera no convenient aquestes dades.
- Integritat: ens indica que els sistemes i les seves dades només poden ser modificats d'una manera controlada.
- Disponibilitat: ens indica que els sistemes i les dades estaran disponibles per a les persones o elements autoritzats.

Aquestes tres característiques han d'existir perquè hi hagi seguretat.

Dins d'aquestes característiques, de vegades es potencien les unes més que les altres. Hi ha algunes que interessen diferents tipus d'empreses o organitzacions, per exemple, un hospital posarà molt interès a la confidencialitat de les dades que tenen els seus sistemes sobre la disponibilitat (preferim que un atacant no accedeixi a les dades que no pas que pugui destruir-les; les dades les podrem recuperar d'una còpia de seguretat).

Aquests departaments o equips no solament han de tenir en compte la seguretat informàtica des del punt de vista de control d'accessos no desitjats als nostres sistemes d'informació. Aquests departaments també han de garantir la integritat i la disponibilitat de les infraestructures i les dades.

Què s'ha de protegir?

Els tres elements bàsics que s'han de protegir en un sistema informàtic són els programes, el maquinari i les dades. Per maquinari ens referim a tots els ele-

ments físics que contenen els sistemes informàtics, com ara servidors, ordinadors personals, commutadors, unitats de cinta… Per programes ens referim al conjunt de programes utilitzats dins de l'empresa i perquè tot funcioni correctament. Les dades es refereixen al conjunt d'informació que desem en els nostres sistemes que utilitzen els nostres programes; per exemple, el contingut d'una base de dades.

Normalment, les dades són l'element principal que s'ha de protegir. És on segurament la nostra empresa basa tot el seu funcionament. És el més difícil de recuperar, un servidor trencat es pot substituir per un altre de nou, un programa es pot reinstal·lar dels CD originals, però les dades s'han de restaurar d'una còpia de seguretat i, segurament, no arribarem a restaurar les dades just en el moment de la pèrdua.

Es poden fer múltiples tipus d'atacs contra qualsevol dels tres elements que hem descrit anteriorment. Però, generalment, es poden identificar de la manera següent:

- Interrupció: és el tipus d'atac que produeix que un sistema es perdi, quedi inutilitzable o no disponible. És un atac a la disponibilitat. Un exemple d'atac és l'atac de denegació de servei (DoS).
- Intercepció: si es produeix un accés no autoritzat a un sistema o a les dades, és un atac a la privadesa. Un exemple pot ser la intercepció de paquets de dades.
- Modificació: si, a més a més, d'un accés no autoritzat tenim modificació de les dades fins i tot la destrucció d'aquestes. Aquest és un atac contra la integritat de les dades. Un exemple pot ser la modificació d'uns fitxers o una base de dades per part d'algú no autoritzat.
- Fabricació: aquest tipus d'atac consisteix a crear un sistema que suplanti l'original. És un atac a l'autenticitat. Per exemple, una suplantació d'un web.

De qui ens hem de protegir?

En aquest apartat parlarem d'allò del que potencialment ens hem de protegir.

Persones

Molts dels atacs que es fan per part de persones, alguns d'involuntaris, altres de voluntaris, poden provocar enormes pèrdues en la nostra empresa. Normalment seran *hackers* o *crackers* que intentin violar la integritat dels nostres sistemes aprofitant un forat de seguretat dels programes. Però hem de tenir en compte que aquests no són els únics que poden provocar pèrdua o fuga d'informació.

Descriurem els diferents tipus de persones que poden constituir un risc per als nostres sistemes:

- Empleats: els empleats de l'empresa rarament es veuen com un problema de seguretat potencial. Es pressuposa que tenen integritat i que se'ls poden confiar les dades de l'empresa. Normalment no es té en compte que gairebé qualsevol persona dins de l'empresa té accés a molta informació i pot comprometre la seguretat dels equips, de vegades, no de manera intencionada, però quan es produeix un problema de seguretat per part d'una persona de l'empresa es produeixen efectes molt perjudicials, ja que aquests coneixen les dades que contenen els sistemes informàtics i com fer més mal. Aquests atacs no han de ser necessàriament de manera lògica, poden ser de manera física. Per aquest motiu, cal tenir format en seguretat i vigilància el personal de l'empresa.
- Exempleats: altres de les persones que potencialment poden ser perilloses per als nostres sistemes són els exempleats i, dins d'aquest grup, els que han abandonat l'empresa de manera voluntària o de manera no correcta. Aquest descontentament pot provocar problemes de seguretat, des de fuga d'informació a esborrament de dades, ja que normalment són persones que coneixen l'entorn.
- *Hackers*: no els hem de confondre amb els *crackers*, els *hackers* són, juntament amb els *crackers,* els atacants més típics dels sistemes. Aquestes persones normalment estan en formació o, ja disposen de grans coneixements, i tenen curiositat pels sistemes. Investiguen la manera d'obtenir privilegis dins de xarxes o veure com estan construïts els sistemes als quals no tenen accés. Encara que no són atacs destructius (com a màxim eliminen les empremtes per no ser detectats) no beneficien els nostres sistemes, ja que poden fer públiques fallades en els nostres sistemes i altres persones sense les

mateixes intencions hi poden accedir i destruir-los o robar-hi dades importants.

- *Crackers*: els *crackers* són persones que es volen aprofitar dels nostres sistemes, ja sigui robant dades, utilitzant els nostres sistemes de passarel·la o per simple diversió. Normalment, ataquen xarxes amb seguretat mitjana, on la seguretat es té poc en compte i de vegades amb pocs coneixements són capaços de vulnerar la seguretat. Aquí es poden englobar des dels principiants que utilitzen eines de *hacking* que troben per Internet fins a verdaders experts.
- Intrusos remunerats: aquestes persones són realment perilloses, encara que no són gaire habituals. Normalment, ataquen grans corporacions i organismes oficials (guerra cibernètica). Es tracta de persones amb grans coneixements en seguretat i en sistemes, que són pagats per fer treballs, des de robar dades fins a modificar-les o eliminar-les.

Programes

Sota aquest tipus, ens trobem els programes instal·lats en el nostre sistema, des de programes instal·lats per nosaltres que tenen errors o *bugs*, fins a programes maliciosos (*malware*) instal·lats per tercers.

Descriurem els diferents tipus de programes que poden constituir un risc per als nostres sistemes:

- Programes incorrectes/*bugs*: les amenaces més comunes en els sistemes provenen d'errors de programa. Aquests errors normalment no són intencionats i són coneguts com a *bugs*. La forma utilitzada comunament per atacar aquests programes són programes anomenats *exploits*, que consisteixen en un programa que aconsegueix normalment donar permisos de superusuari o administració aprofitant el *bug* del programa objectiu. Això és perillós, ja que qualsevol persona amb pocs coneixements pot accedir a sistemes amb *bugs* utilitzant *exploits* i aconseguir el control total de sistemes complets. Per aquest motiu, cal tenir actualitzats els nostres sistemes i tots els programes instal·lats, ja que per culpa d'una badada en un programa que no s'utilitzi ens poden acabar traient el control del nostre sistema.
- Eines de seguretat: les eines de seguretat tenen dos punts de vista: des del punt de vista de l'administrador dels sistemes, que és capaç d'analitzar el

sistema i detectar i arreglar un error de seguretat. Però de la mateixa manera que l'administrador pot fer això, un atacant pot analitzar el sistema, detectar aquestes fallades i aprofitar-les per accedir al sistema. Aquestes eines, com Nessus o Whireshark, poden ser útils en mans dels administradors, tant com en mans d'uns pirates informàtics.

- Portes del darrere / *backdoors*: en els desenvolupaments grans és habitual per part dels programadors programar «dreceres» dins els sistemes d'autenticació. Aquestes dreceres es coneixen com a *backdoors* o portes del darrere i s'utilitzen perquè, a l'hora de desenvolupar grans programes, ens puguem saltar autenticacions per anar més de pressa en la detecció de fallades i en la depuració del programa. El problema d'aquestes dreceres és quan després del desenvolupament no es tanquen, ja que un atacant pot descobrir aquestes portes i accedir als sistemes impunement. També pot ser utilitzat per personal que ja no treballa a l'empresa (exempleats) per accedir a dades sensibles.
- Bombes lògiques: les bombes lògiques són part del codi d'un programa que a priori no realitza res, però que s'activa segons unes condicions o per part d'un tercer. Quan s'activa la bomba lògica, el programa fa accions per a les quals no ha estat creat. En general, aquestes bombes realitzen accions perjudicials o fins i tot l'obertura de *backdoors*. Activadors de les bombes lògiques poden ser dates, creació de fitxers o esborrament de registres d'una base de dades.
- Canals coberts: un canal cobert és un curs de comunicació que permet a un procés receptor i a un d'emissor intercanviar informació de manera que violi la política de seguretat del sistema; essencialment, es tracta d'un mètode de comunicació que no és part del disseny original del sistema, però que es pot utilitzar per transferir informació a un procés o usuari que a priori no estaria autoritzat a accedir a la informació esmentada.
- Virus: un virus és un codi que s'insereix en un fitxer executable, de manera que quan s'executa aquest fitxer infectat, el virus fa alguna acció per a la qual està programat. Aquestes accions poden anar des de la infecció de més executables fins a la destrucció de dades, l'enviament de dades a altres sistemes aliens al nostre, etc.
- Cuc: el terme cuc fa referència a programes capaços de viatjar per si mateixos a través de xarxes de sistemes per fer qualsevol activitat una vegada assolida una màquina. Encara que aquesta activitat no té per què comportar perill, els cucs poden instal·lar en el sistema assolit un virus, atacar aquest

sistema com faria un intrús o, simplement, consumir excessives quantitats d'amplada de banda en la xarxa afectada. Encara que es tracta de *malware*, és moltíssim menys habitual que, per exemple, els virus o les portes del darrere, ja que escriure un cuc perillós és una tasca molt difícil. Els cucs són una de les amenaces que potencialment pot causar més danys.

- Conill/bacteri: aquests programes no representen un dany real a les dades, l'únic que fan és reproduir-se fins a esgotar els recursos del sistema, cosa que genera una denegació de servei. Un exemple típic d'aquest tipus de programa és un bucle que reserva memòria, en cas que sigui un bucle infinit. Aquest esgotarà la memòria del sistema i provocarà una denegació de servei.
- Troians: els cavalls de Troia són codis de programa ocults en programes que utilitza l'usuari i mentre aquests programes realitzen les seves funcions normalment, paral·lelament, realitzen accions ocultes sense el coneixement de l'usuari. Aquests programes normalment modifiquen el comportament de certs programes de seguretat, perquè l'administrador/usuari no s'adoni que està fent accions que no s 'esperen d'ell.

Catàstrofes

Les catàstrofes de qualsevol tipus són normalment l'amenaça que menys probabilitat té de succeir, per exemple, un terratrèmol és difícil que passi en una zona de baixa intensitat sísmica comparat amb la possibilitat de l'atac d'un pirata informàtic buscant dades importants de la nostra empresa. No obstant això, les catàstrofes, encara que tinguin una probabilitat baixa que passin, cal tenir-les en compte, ja que quan passen, provoquen grans danys en les infraestructures de les empreses. Exemples de catàstrofes que contemplarem són terratrèmols, inundacions, incendis, atemptats de baixa magnitud…

Com protegir-nos?

Hem presentat els conceptes que engloba la seguretat, del que volem protegir, de les possibles amenaces que es poden presentar i del seu origen. Ara parlarem de les formes de protegir-nos d'aquestes amenaces. En aquest apartat, presentarem punts que abordarem en temes successius.

Per protegir els nostres sistemes hem d'analitzar les amenaces potencials en els nostres sistemes i la probabilitat que aquestes s'esdevinguin. D'aquesta manera podem dissenyar una política de seguretat per definir responsabilitats i normes per evitar o minimitzar els efectes en cas que passin. Aquesta política de seguretat és la que s'encarregarà de la protecció dels sistemes i les infraestructures de la nostra empresa. Els mecanismes definits en la nostra política de seguretat els anomenarem mecanismes de seguretat.

Aquests mecanismes es divideixen en tres grups:

- Prevenció: els mecanismes de prevenció són els que augmenten la seguretat durant el funcionament normal d'aquest, i prevenen els atacs.
- Detecció: els mecanismes de detecció són els que s'utilitzen per detectar els atacs i les possibles violacions de la seguretat.
- Recuperació: els mecanismes de recuperació són els que aplicarem quan s'ha produït una violació dels sistemes i hem de tornar al funcionament normal d'aquests.

Tenint en compte que els tres mecanismes són importants, hi ha dos en què hem d'emfatitzar la importància dels mecanismes de prevenció i detecció. Com millor sigui la nostra política de prevenció menys haurem d'utilitzar els altres dos mecanismes.

Els mecanismes de prevenció més habituals són:

- Mecanismes d'autenticació i identificació: aquests mecanismes possibiliten la identificació de persones o entitats del sistema d'una manera unívoca i, una vegada realitzat això, autentificar-les. Aquests mecanismes són importants, ja que és la base d'altres mecanismes que basen el funcionament en la identitat per accedir a dades. Uns dels mecanismes més importants d'aquests són els sistemes d'autenticació d'usuaris.
- Mecanismes de control d'accés: totes les dades del sistema han d'estar protegides mitjançant mecanismes de control d'accés, que controlen els tipus d'accés que qualsevol persona té sobre les dades.
- Mecanismes de separació: si els sistemes disposen de diferents nivells de seguretat s'han d'implementar mecanismes que separin uns objectes de diferents nivells, evitant el flux d'informació entre diferents nivells. Els mecanismes de separació es divideixen en cinc grans grups: separació física, temporal, lògica, criptogràfica i fragmentació.

- Mecanismes de seguretat en les comunicacions: és molt important per a la seguretat dels sistemes de la nostra empresa la protecció de la integritat i la privadesa de les dades quan es transmeten a través de les xarxes. Per garantir que la nostra informació es transmet d'una manera segura, s'han d'utilitzar mecanismes que normalment es basen en la criptografia: xifratge de clau pública, de clau privada, signatures digitals, protocols segurs com Openssl, https. Aquest és un punt molt important en el disseny d'aplicacions web, ja que manté la integritat de les dades dels nostres clients.

Passos per definir un pla de seguretat

En els apartats anteriors hem vist què és la seguretat, que ens hem de protegir i com ens podem protegir. En aquest apartat veurem com definir un pla de seguretat tenint en compte els apartats anteriors.

Els passos que s'han de seguir per definir un pla de seguretat són els següent:

a) Predir els possibles atacs i analitzar-ne els riscos: la primera fase de la metodologia és determinar els possibles atacs que es poden esperar i la forma de defensar-se'n. No és possible estar preparats per a tots els atacs, per aquest motiu, analitzarem de tots els possibles atacs els que són més probables que tinguin lloc. Sempre és millor prevenir o atenuar l'atac que reparar el dany que s'ha causat. La predicció dels atacs intenta pronosticar-ne la probabilitat, cosa que depèn del coneixement dels seus diferents aspectes.

b) Per a cada tipus d'amenaça: considerant totes les amenaces possibles que causen atacs en els nostres sistemes, establirem dos tipus d'estratègies per a cada amenaça: proactives i reactives.

1. Estratègia proactiva: l'estratègia proactiva és un conjunt de passos predefinits que s'han de seguir per evitar atacs abans que passin. Entre aquests passos s'inclou observar com podria afectar o danyar el sistema i els punts vulnerables. Els coneixements adquirits en aquestes avaluacions poden ajudar a implementar les directives de seguretat que aturessin o disminuïssin els atacs.

Els tres passos de l'estratègia proactiva són:

- Determinar el dany que causaria l'atac: els danys poden variar des de petites fallades de l'equip fins a la pèrdua de dades. El dany causat al sistema dependrà del tipus d'atac. Sempre que es pugui, provarem els danys en un entorn de prova per aclarir els danys que puguin provocar els diferents tipus d'atacs.
- Establir els punts vulnerables i les debilitats que explotaria l'atac: si es poden descobrir els punts vulnerables que explota un atac específic, es poden modificar les normes i els controls de seguretat actuals o implementar-ne altres de noves per reduir aquests punts vulnerables. La determinació del tipus d'atac, amenaça i mètode facilita el descobriment dels punts vulnerables existents. S'han de determinar els punts vulnerables o debilitats en les àrees de seguretat física, de dades i de xarxa.
- Reduir els punts vulnerables i les debilitats que s'han determinat en el sistema per a aquest tipus d'atac específic: la reducció dels punts vulnerables i de les debilitats del sistema de seguretat que s'han de determinar en l'avaluació anterior és el primer pas per desenvolupar directives i controls de seguretat eficients. Aquesta és la compensació de l'estratègia proactiva. Mitjançant la reducció dels punts vulnerables, es poden disminuir tant la probabilitat de l'atac com de la seva eficàcia. Un problema d'introduir molts controls de seguretat pot provocar que accedir a unes dades per part d'una persona pugui ser molt tediós i complicat. Per exemple, podríem tancar un servidor en una habitació amb accés restringit, sense accés a la xarxa, aquest seria un servidor gairebé inexpugnable, però la persona que accedeix a les dades hauria realment d'accedir-hi i per aquest motiu s'ha d'arribar a un compromís entre controls de seguretat i accessibilitat.
- El seguiment d'aquests passos per analitzar els diferents tipus d'atacs ens serà útil per determinar les àrees de vulnerabilitat que plantegen el risc més elevat per a l'empresa. També cal mesurar el cost dels danys econòmics que ens poden produir aquests atacs.
- Aquests plans no són totalment efectius, sempre hi ha la possibilitat d'un *bug* en un programa no corregit o ocult. Per aquest motiu cal desenvolupar plans de recuperació (còpies de seguretat) i plans de contingència per als casos que no puguem controlar.

2. Estratègia reactiva: l'estratègia reactiva s'implementa quan ha fallat l'estratègia proactiva i defineix els passos que s'han d'adoptar després de l'atac o

durant un atac. Ajuda a identificar el dany causat i els punts vulnerables que es van explotar en l'atac, a determinar per què es va produir, reparar el dany que va causar i a implementar un pla de contingència. Tant l 'estratègia reactiva com la proactiva funcionen conjuntament per desenvolupar normes i controls de seguretat amb la finalitat de reduir els atacs i el dany que causen als sistemes.

c) Avaluar el dany: determinar el dany causat durant l'atac. Això s'ha de fer com més aviat millor perquè les restauracions també comencin com més aviat millor. Si no és possible l'avaluació dels danys, haurem de recórrer als plans de contingència perquè es pugui prosseguir amb les operacions normals de l'empresa.

d) Determinar la causa del dany: determinarem la causa del dany. Cal saber a quins recursos anava dirigit l'atac i quins punts es van explotar perquè tingués èxit. Això es coneix com a anàlisi forense.

e) Reparació de dany: és important que es repari el dany causat per l'atac com més aviat millor. Els plans i procediments de recuperació i/o contingència han de respondre a aquest punt.

f) Documentar i aprendre: és important documentar tots els atacs que es produeixin. La documentació generada ha d'abraçar tot el que ha passat en l'atac, l'objectiu de l'atac, com ha succeït i com ens hem recuperat.

g) Revisar el resultat i fer simulacions: després dels atacs, revisarem el resultat d'aquests. Si és possible, durant l'atac, també és convenient que es realitzi un seguiment per veure els punts del nostre sistema que s'estan atacant i veure si tenim implantats algun sistema de seguretat.

h) Revisar l'eficàcia de les mesures: si s'han produït atacs, revisar si les nostres mesures de seguretat han pogut aguantar. En cas contrari, veure on han fallat aquestes mesures.

i) Ajustar les mesures: si les mesures no han estat suficients, revisarem les nostres mesures i el nostre pla de seguretat per afegir-ne de noves, o modificarem mesures perquè els atacs que han tingut èxit no es tornin a produir.

3.2. Seguretat dels serveis web

En els apartats següents analitzarem capa per capa i es donarà rellevància als requeriments de seguretat a cadascun d'ells, els mètodes per garantir la seguretat i els procediments d'actuació davant d'un error de seguretat.

Es presentaran els punts següents respecte de la seguretat:

- Seguretat física.
- Personal.
- Autenticació.
- Programes.
- Seguretat del sistema.
- Seguretat en bases de dades.

També es presentaran els temes d'auditories de seguretat i els tipus d'atacs que es poden produir als sistemes.

3.3. Seguretat física

Introducció

La seguretat física en els sistemes informàtics consisteix en l'aplicació de barreres físiques i procediments de control com a mesures de prevenció i contramesures contra les amenaces als recursos i a la informació confidencial. És a dir, són els mecanismes destinats a protegir físicament qualsevol recurs del sistema.

La seguretat física no és un tema en què s'aprofundeixi gaire dins de la seguretat informàtica. Moltes vegades, els departaments estan més ocupats en els atacs dirigits al web de l'empresa que en la seguretat d'accés al seu CPD. Això motiva que en algunes situacions un atacant decideixi aprofitar vulnerabilitats en la seguretat física en lloc de vulnerabilitats lògiques, ja que possiblement sigui més fàcil accedir als dispositius físics. La seguretat física no pot ser un tema oblidat i se li ha de donar la importància que té: un robatori o una catàstrofe que afecti el centre de processos de dades pot ser molt més perjudicial que un atac a un *bug* del tallafocs (*firewall*).

Protecció del maquinari

El maquinari és un dels elements més cars dins dels sistemes de les empreses, per aquest motiu és convenient protegir-lo correctament.

En els punts següents presentarem les amenaces més comunes en la protecció del maquinari.

Control d'accés físic

La capacitat d'accedir a un sistema físicament fa inútil qualsevol altra mesura de seguretat que tinguem. Hem de pensar que si un atacant pot arribar físicament on vol estem perduts, ja que es pot emportar cintes, discos, servidors sencers. Pot apagar els sistemes, arrencar-los amb versions alternatives del sistema operatiu… és a dir estem perduts.

Un cop dit això, tenim clar que cal tenir un control d'accés als sistemes físics de l'empresa per tal de garantir-ne la seguretat global. El nivell de seguretat dependrà de l'entorn on estiguin ubicats els sistemes. Un servidor pot estar protegit en un CPD amb vigilància, però podem tenir components de xarxa (*switches*, *hubs*) repartits per les plantes de la nostra empresa sense cap tipus de seguretat implementada.

Per aquests motius hem de tenir en compte la prevenció i la detecció.

- Prevenció: com podem impedir un accés no autoritzat? Hem de tenir en compte que hi ha múltiples mètodes de control d'accés: analitzadors d'empremta dactilar, targetes d'accés, simples claus o seguretat privada. Aquests mètodes de control d'accés es poden barrejar segons el que hem de protegir, per exemple, amb seguretat a l'entrada de l'edifici, control mitjançant targetes d'accés per accedir als CPD… En definitiva, el que tractarem és evitar que qualsevol individu aliè a les instal·lacions pugui accedir-hi lliurement.
- Detecció: si la prevenció no és possible, hem de passar a la detecció, mitjançant cambres de circuit tancat o alarmes amb mecanismes tradicionals de detecció d'intrusos en les instal·lacions. També és important conscienciar els empleats, que han d'estar atents a persones alienes a l'empresa i, si cal, s'han de posar en contacte amb seguretat de l'edifici si detecten que una persona aliena a l'empresa està accedint a sistemes informàtics sense autorització.

Desastres naturals

Aquests problemes no solen ser habituals, però en cas que es produeixin, causin greus danys en les infraestructures. Encara que no siguin habituals, s'ha de tenir previsió del que hem de fer o el pla que tenim en cas que es produeixi un desastre natural.

Podem identificar els desastres naturals següents:

- Terratrèmols.
- Tempestes elèctriques.
- Inundacions i humitat.

Per a aquests desastres naturals haurem d'implementar mecanismes de seguretat per prevenir cadascun d'aquests. Per exemple, per a les tempestes elèctriques, cal instal·lar parallamps, o per a inundacions, s'ha de dissenyar l'edifici de manera que es pugui protegir d'aquestes. Encara que cal tenir en compte que els millors mecanismes contra els desastres naturals són els plans de contingència amb centres de suport (sempre que aquests estiguin a una distància considerable).

Desastres de l'entorn

Aquests problemes són més comuns que els desastres naturals i, per aquest motiu, hem d'anar més compte amb ells. Els desastres de l'entorn més comuns són els següents:

- Electricitat.
- Soroll elèctric.
- Incendis i fum.
- Temperatures extremes.

Per a aquests tipus de desastres, tindrem sistemes relacionats amb la construcció dels centres de procés de dades.

Per evitar problemes amb l'electricitat, tindrem sistemes d'alimentació ininterrompuda (SAI) que ens permetran poder fer parades controlades dels nostres sistemes d'una manera ordenada, en cas que se'n vagi la llum. Si cal seguir operant amb els nostres sistemes en cas d'apagada, caldrà contractar una segona línia de llum o, fins i tot, la instal·lació d'un generador elèctric dièsel.

Per als incendis i el fum, s'han d'instal·lar sistemes secs contra incendi i sistemes de ventilació pensant en els gasos.

Per a les temperatures extremes (normalment considerarem les temperatures altes), s'han d'instal·lar sistemes de refrigeració. Aquests sistemes poden ser des de simples sistemes d'aire condicionat fins a nous sistemes que consideren la ubicació de la construcció del centre de processos de dades en llocs on es pugui utilitzar la baixa temperatura exterior o l'aigua del mar per refrigerar els sistemes.

Protecció de les dades

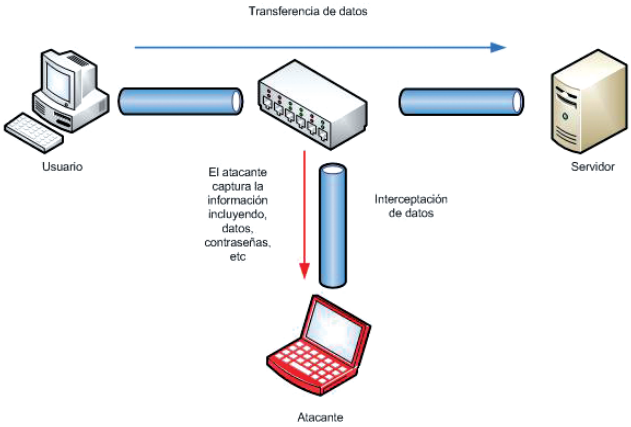
La seguretat física també comporta la protecció de les dades dels nostres sistemes, és a dir, tant les emmagatzemades en els nostres sistemes com les que es transfereixen per les xarxes. Tenint en compte que tota la protecció física correspon a una protecció de les dades, hem de tenir en compte certs aspectes a l'hora de dissenyar les nostres polítiques de seguretat físiques. Aquesta protecció física ha de tenir en compte tant la protecció dels sistemes físics com els accessos per aconseguir la informació emmagatzemada en els sistemes.

Intercepció passiva

La intercepció passiva o *eavesdropping* és un procés mitjançant el qual un ens captura una informació de qualsevol tipus que no va dirigida cap a ell; aquesta intercepció es pot fer de diferents maneres, des de la instal·lació d'un agent en un PC o servidor que està ubicat dins d'una xarxa fins a algun intrús que ha aconseguit accedir a un *switch* de l'empresa i ha punxat algun dispositiu d'escolta, etc. Encara que és un atac passiu, és difícil de detectar, ja que no efectuen cap canvi dins dels nostres sistemes, simplement, escolta tota la informació que passa per la xarxa, contrasenyes, informació secreta, informació bancària, sense que ningú se n'adoni. Aquest atac pot ser la bestreta d'un atac actiu.

El format més usual a l'hora d'interceptar informació és l'*sniffing*, que consisteix a capturar trames que es transmeten per la xarxa mitjançant un programa que s'està executant en un PC o servidor connectat a la xarxa, o algun tipus de dispositiu que es connecti al cablejat de xarxa, per exemple en la il·lustració següent tenim un atacant connectat amb un portàtil en un *switch* de l'empresa escoltant tot el trànsit que hi ha entre els clients i el servidor.

Aquests atacs són difícils de detectar. La manera d'evitar aquest tipus d'atacs és, per començar, impedint que sigui difícil d'accedir al cablejat de xarxa i als



Intercepció de dades

dispositius de xarxa, encara que això és complicat, ja que si una persona no autoritzada accedeix a les instal·lacions pot desconnectar un PC de la xarxa i connectar-hi el seu dispositiu. Per solucionar això, podem utilitzar programes que revisen la xarxa buscant dispositius en mode promiscu, que és un dels indicatius que hi ha en un programa d'*sniffer* en funcionament.

Un altre tipus de solucions passa per utilitzar dispositius de maquinari o programes de xifratge punt a punt. És una solució cara i és difícil d'implementar, ja que hauríem de posar dispositius entre totes les connexions. És una solució utilitzada quan hem de connectar dues oficines que estan ubicades en dues localitzacions diferents i posaríem aquests dispositius a l'entrada de cadascuna de les xarxes.

Còpies de seguretat

Encara que la implantació de sistemes de còpies de seguretat la tractarem àmpliament en temes posteriors, en aquest apartat plantejarem les còpies de seguretat dins d'un pla de seguretat.

Si finalment un atac als nostres sistemes té èxit i el que busca és esborrar dades o modificar informació, l'única manera que tindrem de tornar a la situació anterior és gràcies a les nostres còpies de seguretat. També haurem de tenir en compte que pot ser la salvació de la nostra empresa si es produeix un desastre natural i la majoria dels nostres sistemes resulten danyats i són irreparables.

També és possible que un atac consisteixi a accedir al magatzem de suports de còpies per robar aquests suports i accedir a la informació que aquests continguin.

Per aquest motiu, en un pla de seguretat ha d'estar definida la seguretat a la qual estaran sotmesos els dispositius de còpia i els suports on aquestes es fan, i s'hi ha de tenir en compte:

- On emmagatzemar els suports de còpia: haurem de tenir en compte que no podem emmagatzemar tots els suports de còpia de seguretat en el mateix lloc on tenim els sistemes, encara que ens resulti molt còmode a l'hora de fer les restauracions. Per començar, les còpies de seguretat s'hauran d'emmagatzemar en algun tipus d'armari ignífug i que també suporti la humitat i, addicionalment, haurem d'emmagatzemar certs suports, com per exemple, els jocs de cintes que no estiguem utilitzant actualment, les còpies setmanals, en una ubicació física allunyada del CPD. Per portar a terme això, podem utilitzar una altra seu en una altra ubicació de l'empresa o contractar empreses de seguretat que tenen serveis d'emmagatzematge de còpies de seguretat, on garanteixen tenir les còpies de seguretat en llocs preparats expressament (ignífugs, preparats contra inundacions i humitat, etc.).
- Seguretat d'accés als suports de còpia: les còpies de seguretat emmagatzemen totalment la informació de l'empresa. Per aquest motiu, hem de tenir molt en compte qui pot accedir a aquests suports i dispositius. Per a això, haurem d'emmagatzemar les còpies de seguretat en llocs amb alguna mesura de seguretat, encara que sigui un simple pany, amb una llista identificada de persones que poden accedir a aquestes còpies. Hem de tenir en compte que l'accés a la informació que contenen aquestes còpies pot fer que una altra empresa ens robi dissenys o informació estratègica de la nostra empresa i ens porti a la fallida.
- Identificació dels suports: cal ser curós en la identificació de les còpies de seguretat, amb això ens referim que és molt pràctic etiquetar un suport amb «Base de dades de producció», però això no és l'idoni, ja que una persona aliena a l'empresa pot identificar sense problemes el que conté aquesta còpia de seguretat. El millor és establir codis que identifiquin el suport dins d'un joc de suport (per exemple L1D, indicant que és el suport de dilluns, del joc 1, que és diària) i identificant el que conté cada suport en un fitxer, o simplement utilitzant el catàleg del programa per fer les còpies de seguretat.

Aquests són els punts que haurem de tenir en compte de les còpies de seguretat dins del pla de seguretat de la nostra empresa.

3.4. Personal

Introducció

Dins de la seguretat informàtica, un dels punts més febles, si no el més feble, és el personal que accedeix als sistemes informàtics de l'empresa. Totes les persones de l'empresa són susceptibles de cometre errors o de no ser prou rigoroses amb la seguretat dels sistemes informàtics. Un administrador de sistema pot deixar un fitxer sense protegir o una persona de seguretat pot deixar entrar lliurement personal aliè de l'empresa. Aquests són exemples de problemes de seguretat que poden provocar les persones.

Dins dels problemes de seguretat que poden provocar les persones, pot passar que que no tots siguin atacs a la seguretat del nostre sistema; de vegades, es produeixen problemes de seguretat a causa de badades, falta d'atenció o simplement deixadesa. Per aquest motiu, s'ha d'inculcar el tema de la seguretat informàtica a tot el personal de l'empresa.

Tipus d'atacs

Enginyeria social

L'enginyeria social consisteix en la manipulació de persones perquè voluntàriament facin actes que normalment no farien. Ningú no vol que el manipulin, però de vegades es dóna la situació que un atacant pot manipular una persona perquè faci alguna cosa. Aquest tipus d'atacs és dels menys perillosos per a l'atacant i dels més efectius.

Un típic atac d'aquest estil són els correus que demanen que posis una contrasenya nova en algun sistema, amb un valor determinat, per fer proves, normalment signada per l'administrador. Aquest tipus d'atacs són fàcils d'emmascarar i, probablement, un usuari que no ha estat entrenat en seguretat informàtica farà aquest canvi i donarà a l'atacant la possibilitat d'accedir al sistema de manera fàcil i sense haver de «tacar-se les mans».

La primera solució, i la més senzilla, per a aquests problemes és la següent: un administrador de sistemes mai no us demanarà la contrasenya, ni mai no us demanarà que poseu una contrasenya determinada. Cal ignorar qualsevol mena

d'aquestes peticions, encara que siguin per correu electrònic. L'administrador de sistemes té altres opcions per provar coses. És simple, però és la solució més efectiva. Cal educar els usuaris amb aquesta norma.

Shoulder surfing

Consisteix a espiar físicament alguna informació de l'usuari, normalment contrasenyes i usuaris. Un exemple d'això és obtenir les contrasenyes dels usuaris que utilitzen papers per apuntar les seves contrasenyes i les deixen enganxades al monitor o a sota del teclat. Qui vulgui hi pot tenir accés. És habitual en molts llocs i l'única solució és conscienciar els usuaris que no és un comportament correcte.

No solament són els usuaris les víctimes del *shoulder surfing*. Hi ha programes en què s'ha d'introduir una contrasenya i aquesta no s'oculta, surt en clar. Qualsevol que estigui situat a prop pot obtenir la contrasenya que estem teclejant.

Suplantació d'identitat

La suplantació d'identitat o *masquerading* consisteix en la suplantació de la identitat d'una persona. Normalment es busca suplantar un usuari autoritzat a entrar en els sistemes informàtics. Aquesta suplantació pot ser de manera electrònica o física.

Aquest atac, en el seu format físic, no és habitual en entorns normals, ja que en general seran àrees d'accés restringit i controlat pel mateix personal de l'empresa. En aquest cas, un atac d'aquest tipus no sol ser efectiu, ja que és molt fàcil detectar l'intrús perquè es tracta d'àrees dins les quals tot el personal «habitual» es coneix. La suplantació d'identitat és més corrent en entorns on hi ha controls d'accés físic, i on un intrús pot «enganyar» el dispositiu –o la persona– que realitza el control, per exemple, amb una targeta d'identificació robada que un lector accepta o amb un carnet falsificat que un guàrdia de seguretat dóna per bo.

Una variant del *masquerading* el constitueix l'atac denominat *piggybacking*, que consisteix simplement a seguir un usuari autoritzat fins a una àrea restringida i accedir-hi gràcies a l'autorització atorgada a aquest usuari (en la nostra vida personal, es relaciona amb la cura que cal tenir de no deixar passar ningú pel nostre portal, encara que aparentment estigui buscant les claus per obrir).

Contra això s'han d'aplicar les mateixes mesures que contra la mascarada física: controls d'accés estrictes i convenientment verificats. Els exemples de *piggybacking* són molt habituals: des d'un atacant que es vesteix amb un mono de treball i que càrrega amb equips a la porta d'una sala d'operacions, per accedir-hi al costat d'un usuari autoritzat a una altra modalitat d'aquesta tècnica, que es relaciona amb els reconeixedors de targetes intel·ligents que obren la porta d'una sala però que, una vegada oberta, no es preocupen de comptar quantes persones la travessen.

Atacs interns

Dins dels atacs relacionats amb les persones, els més perillosos són els relacionats amb els atacs que fan les mateixes persones.

La majoria dels fraus, robatoris, sabotatges o accidents dels sistemes d'informació de les empreses són realitzats pel propi personal de l'empresa. Per aquest motiu, hem de tenir en compte que molts dels atacs que es produiran als nostres sistemes vindran des de dins, de persones que treballen en l'empresa o que hi han treballat. Moltes de les persones que treballen en l'empresa coneixen els mètodes de seguretat, els punts febles i les aplicacions més crítiques de l'empresa (coneixen l'aplicació que és més crítica de l'empresa i poden esborrar dades estratègics de l'empresa).

Per què es poden produir aquests atacs? Diners, xantatges, gent descontenta.... Qualsevol d'aquests motius pot ser suficient perquè una persona faci un atac als sistemes de l'empresa, sigui en forma de robatori d'informació, esborrament de dades, modificació de dades sense autorització.

Com enfrontar-nos a aquests problemes?

Les solucions als problemes de seguretat relacionats amb el personal són dels més difícils de solucionar.

De vegades no és fàcil convèncer la gent que no pot deixar la contrasenya apuntada en un paper enganxat al monitor. S'ha de conscienciar la gent de quines són les seves obligacions i de la importància de la seguretat informàtica.

A més a més de conscienciar el personal de l'empresa, també l'hem de formar per aconseguir un sistema de seguretat que funcioni correctament.

En definitiva, el nostre pla de seguretat ha de tenir una part relacionada amb la conscienciació i formació dels empleats de l'empresa.

Respecte dels atacs interns, haurem de prevenir-los tenint en compte els punts següents:

- Mínim privilegi, cada usuari ha de tenir els mínims privilegis per desenvolupar el seu treball de manera normal. És a dir, si un usuari administratiu ha d'accedir a una taula de la base de dades de personal on estan els sous i no volem que accedeixi a aquesta informació, s'ha de crear una vista o consulta especial perquè no pugui accedir a aquesta informació.
- Dins dels sistemes informàtics hi ha certes funcions que són crítiques; per exemple, el coneixement de les contrasenyes d'administració de base de dades. Aquestes funcions s'han de compartir entre diverses persones, perquè en cas que aquesta persona deixi l'empresa, pugui seguir el funcionament normal de l'empresa i perquè es puguin auditar entre elles les accions que estan realitzant.
- Separació i rotació de funcions, no és recomanable que una sola persona posseeixi tota la informació sobre la seguretat d'una empresa. Per aquest motiu és convenient separar les diverses funcions de seguretat de l'empresa i anar-les rotant entre diferents persones, amb l'objectiu que s'auditin entre elles i no acumulin «gaire poder» amb el qual puguin ocultar fets delictius.

3.5. Autenticació

Introducció

En els sistemes d'informació d'una empresa cal poder identificar cada persona que accedeix a aquests sistemes. Aquests sistemes d'autenticació no solament busquen identificar les persones, sinó també autenticar que la persona diu qui és.

Els mètodes d'autenticació són els següents:

- Sistemes basats en alguna cosa coneguda. Per exemple, un *password* (Unix) o *passphrase* (PGP).
- Mètodes basats en alguna cosa posseïda. Per exemple, una targeta d'identitat, una targeta intel·ligent (*smartcard*), un dispositiu USB de tipus epass token, smartcard o dongle criptogràfic.

- Mètodes basats en una característica física de l'usuari o en un acte involuntari d'aquest. Per exemple, verificació de la veu, d'escriptura, d'empremtes, de patrons oculars.

Els sistemes d'autenticació o identificació han de tenir algunes característiques:

- Ser fiables, és a dir, amb taxes d'error molt baixes.
- Si són econòmicament viables per a l'empresa, cal posar lectors d'empremtes dactilars en tots els llocs de treball, encara que pot ser realment car.
- Han de suportar atacs dirigit a ells.
- Han de ser acceptables per als usuaris que els utilitzin. No podem convertir en un treball tediós l'autenticació d'usuaris.

Sistemes basats en alguna cosa coneguda

El model d'autenticació més conegut és el mètode d'autenticació per contrasenyes. És el més bàsic, però continua sent el més utilitzat per entrar al correu electrònic, als nostres comptes bancaris, etc. Aquestes contrasenyes són conegudes per l'usuari i a priori només la sap el. Però aquesta és força vulnerable; per exemple, en una targeta de crèdit, si no hi hagués un màxim d'intents fallits abans d'anul·lar la targeta, un lladre podria acabar encertant el pin/contrasenya que hi dóna accés, només és qüestió de temps i de ser metòdic en la inserció de contrasenyes consecutives.

Tots els sistemes de contrasenyes funcionen de la mateixa manera. De primer, s'estableix una contrasenya amb la qual l'usuari accedirà. Quan l'usuari vulgui accedir al sistema, el sistema li demana l'usuari i la contrasenya. Si l'usuari ho fa correctament, accedirà al sistema.

Aquests sistemes són fràgils, ja que si es disposa de temps indefinit sempre es podran encertar, però, en alguns casos, el temps és tan elevat (de l'ordre d'anys) que no són possibles aquests tipus d'atacs. També es pot donar el cas d'un usuari despistat que apunti la contrasenya en un paper, la comparteixi amb un tercer o simplement que algú vegi com la tecleja.

Sistemes basats en alguna cosa posseïda

El sistema tradicional d'alguna cosa posseïda són les targetes intel·ligents, que són dispositius semblants a les targetes de crèdit que disposen d'un xip en què, normalment, tenen encastat un sistema de fitxers de xifratge i funciones criptogràfiques.

Quan un usuari que posseeix una targeta intel·ligent vol autenticar-se en un sistema, de primer ha d'introduir la targeta en un lector de targetes. Es produeix un intercanvi d'informació entre la targeta i el lector de targetes, per identificar-se correctament. Després, es teclejarà una contrasenya que, si és correcta, autenticarà correctament l'usuari.

Els avantatges d'utilitzar targetes intel·ligents com a sistema d'autenticació d'usuaris són molts. Es tracta d'un sistema que els usuaris accepten bé, ja que només han de recordar una contrasenya i és més segur que el sistema de contrasenyes, ja que disposem d'un dispositiu altament segur.

Els inconvenients poden ser que hem de tenir en compte el cost de la implantació del sistema i que no són infal·libles. Amb lectors de targetes i enginyeria inversa es pot obtenir informació d'aquestes.

Sistemes biomètrics

A part dels sistemes criptogràfics d'autenticació, hi ha una altra classe de sistemes en què no s'aplica la criptografia. Aquests sistemes d'autenticació es basen en la biometria de la persona, en característiques físiques de l'usuari que s'ha d'identificar.

En aquests sistemes la criptografia s'utilitza en un terme secundari, xifrant les comunicacions o les bases de dades que contenen la informació de les empremtes dactilars o els patrons de la retina.

L'autenticació basada en característiques físiques existeix des de fa molt temps. Nosaltres mateixos tenim documents d'identitat en què figura la nostra empremta dactilar unívoca, o simplement es reconeix la gent per la veu, la cara, etc.

La identificació mitjançant mètodes biomètrics pot incloure qualsevol part del cos, però la realitat és que només s'utilitzen els següents:

- Reconeixement del patró de l'iris de l'ull.
- Reconeixement del patró de la retina de l'ull.

- Reconeixement de les empremtes dactilars.
- Reconeixement de la geometria de la mà.
- Reconeixement de la signatura (escriptura).
- Reconeixement de veu.

Cadascun d'aquests tipus d'identificació té uns costos diferents, són més fàcils d'implementar o són més efectius.

El procés general d'autenticació d'un sistema biomètric consisteix en les fases següents:

- Captura o lectura de les dades que l'usuari presenta a validar.
- Extracció de certes característiques de la mostra.
- Comparació de les característiques amb la informació emmagatzemada.
- Decisió de si és vàlid o no.

Dins d'aquests sistemes hi ha la possibilitat dels falsos negatius o falsos rebutjos, que consisteix en el rebuig dels usuaris legítims, ja que no s'aconsegueix identificar l'usuari de manera correcta, i també existeix el contrari, falsos positius o falsa acceptació, que és identificar correctament usuaris il·legítims. Encara que aquesta possibilitat existeixi, és tan baixa que normalment es menysprea en aquests sistemes, ja que cada vegada són més efectius, i és en els falsos positius en què més es treballa per tal de ser minimitzats.

3.6. Programes

Introducció

Els programes tenen fallades i errors de disseny. No és que els programadors ho facin voluntàriament, però de vegades sí que hi ha badades o males pràctiques de programació. Aquests fallades són una de les fonts d'amenaça que té tot sistema informàtic. No solament els errors de programes són problemàtics, sinó que hi ha programes l'objectiu dels quals són comprometre la seguretat. Aquests són els virus, els troians, etc. Aquest tipus de programes no solament comprometen la seguretat quan afecten els administradors, també quan afecten els usuaris es poden produir suplantacions d'identitat, troians que controlin el PC de l'usuari, infecció de fitxers compartits, etc. Per evitar això, només hi ha una mesura de

protecció: la prevenció. És molt important que tots els programes comprovin la font d'on provenen abans d'executar-los, ja que, si no sabem d'on provenen, poden contenir virus.

Errors en els programes

Els errors o *bugs* en els programes constitueixen una de les amenaces a la seguretat que més preocupa els encarregats de la seguretat informàtica. En la majoria de casos no és voluntària la creació de programes amb errors, però és impossible no equivocar-se en programes que tenen milers de línies de codi.

A part dels problemes de seguretat que es poden produir pels errors en els programes, és molt important saber els usuaris amb els quals s'està executant aquest programa. Per exemple, si un servidor web amb un error s'executa amb l'usuari HTTP no té les mateixes conseqüències que si s'executa amb l'usuari administrador de l 'equip. El problema de seguretat del programa de servidor web, en teoria, nomes pot afectar fitxers i processos de l'usuari HTTP (una excepció a això són *bugs* en els programes que permetin escalada de privilegis). La pràctica d'executar aplicacions amb usuaris no administradors és un dels millors mètodes perquè els problemes de seguretat en les aplicacions no afectin altres aplicacions o tinguin conseqüències més importants.

Buffer overflows

És un dels errors més comuns i el més utilitzat és el desbordament de pila o *stack smashing*, també conegut com a *buffer overflow*. Encara que cada vegada els programes són més segurs, especialment els que permeten l'execució com a administrador, és gairebé segur que un atacant que accedeixi al sistema pugui aconseguir privilegis d'administrador mitjançant un *buffer overflow.*

La idea del desbordament de pila és senzilla. Alguns programes permeten corrompre la pila d'execució escrivint més enllà dels límits d'un *array* declarat com a auto. Això pot causar que l'adreça de retorn de la funció esmentada sigui una adreça aleatòria. Això, unit a alguns dels permisos dels fitxers executables (que permeten l'execució com a administrador), fa que el sistema li atorgui privilegis d'administrador a un usuari sense privilegis.

On trobem el problema? El problema resideix en els executables que ens permeten ser executats com a administrador, és a dir, que quan l'executem com a usuari sense privilegis, ens escala permisos d'administrador durant l'execució d'aquest programa, i també el que s'ha inclòs en l'adreça de retorn de la funció problemàtica. Si aquesta adreça de retorn és una *shell*, tindrem una *shell* amb privilegis d'administrador.

Hi ha nombrosos *exploits* (programes que són capaços d'aprofitar els *buffer overflows* dels programes) a Internet i inclouen el codi per executar una *shell* una vegada aprofitat el *bug*.

Per evitar aquesta problemàtica, hem de tenir en compte els punts següents:

- Tenir el mínim nombre de fitxers executables amb la capacitat d'executar-se com a administrador (fitxers amb *setuid*).
- Mantenir els programes i el sistema operatiu el més actualitzat possible. No serveix de res tenir les millors mesures de seguretat perquè ningú no hi pugui entrar, però una badada d'un usuari amb una contrasenya feble pot permetre a un atacant entrar, utilitzar un executable no actualitzat que tingui aquest error i que no hàgim actualitzat.

Virus

Un virus és un codi que s'insereix en un fitxer executable, de manera que quan s'executa aquest fitxer infectat, el virus fa alguna acció que està programada. Aquestes accions poden portar la infecció de més executables, la destrucció de dades, l'enviament de dades a altres sistemes aliens al nostre, etc. El virus necessita obligatòriament un programa on inserir-se per tal d'executar-se, per la qual cosa no el podem considerar un programa o procés a part.

Cal tenir en compte que la majoria de virus afecten sistemes d'escriptori, com els ordinadors personals, o en general els sistemes operatius de la família del Microsoft Windows. Els sistemes operatius dels entorns servidor com els UNIX/Linux (AIX, Solaris, HP-UX, Linux) no tenen virus, ja que, atesa la forma que els usuaris tenen d'entrar a aquests sistemes, sempre sense privilegis, només poden afectar els seus fitxers i els fitxers que tinguin permisos. A causa d'aquesta característica, els sistemes UNIX tenen poc interès per als creadors de virus.

Les mesures que s'han de tenir en compte contra els virus informàtics passa per tenir sistemes antivirus en els ordinadors dels usuaris, sistemes antivirus en

els servidors Windows i, si disposem d'un servidor de correu electrònic, un sistema d'escaneig de virus per a aquest sistema.

Cucs

El terme *cuc* fa referència a programes capaços de viatjar per si mateixos a través de xarxes de sistemes per fer qualsevol activitat una vegada assolida una màquina. Encara que aquesta activitat no ha de comportar necessàriament un perill, els cucs es poden instal·lar en el sistema assolit per un virus, atacar aquest sistema com faria un intrús o, simplement, consumir excessives quantitats d'amplada de banda en la xarxa afectada. Encara que es tracta de *malware* moltíssim menys habitual que, per exemple, els virus o les portes del darrere, ja que escriure un cuc perillós és una tasca molt difícil, els cucs són una de les amenaces que potencialment poden causar més danys.

Contra els cucs és difícil defensar-se, ja que aprofiten errors de seguretat en els programes. La millor solució a aquest problema és tenir actualitzats tots els programes i el sistema operatiu. D'altra banda, és molt important estar assabentat dels errors de seguretat que van sortint.

Conill/bacteri

Aquests programes no comporten un dany real a les dades. El que fan és reproduir-se fins a esgotar els recursos del sistema, fet que produeix una denegació de servei. Un exemple típic d'aquest tipus de programa és un bucle que reserva memòria. Si és un bucle infinit, aquest esgota la memòria del sistema i provoca una denegació de servei.

Un exemple típic d'aquest tipus de programes en C és el següent:

```
main(){
        while(1){
                malloc(1048576);
                fork();
        }
}
```

Aquest programa el que fa és reservar 1 MB en cada pas del bucle i, a més a més, crea una còpia d'aquest en cada pas del bucle. S'executarà contínuament i, en un moment, ens consumirà tota la memòria del sistema.

La millor manera d'evitar aquest tipus d'atacs és utilitzar els sistemes de limitació de recursos que ens ofereixen els sistemes operatius per usuari o per procés. Així, si un usuari només pot consumir un percentatge de la memòria, no ens podrà consumir tota la memòria del sistema amb l'exemple anterior.

Troians

Els cavalls de Troia són codis de programa ocults en programes que utilitza l'usuari i que aquests programes realitzen les seves funcions normalment, encara que paral·lelament farà accions ocultes sense el coneixement de l'usuari. Aquests programes modifiquen normalment el comportament de certs programes de seguretat, perquè l'administrador/usuari no s'adoni que està fent accions que no s'esperen d'ell.

El mètode més efectiu per descobrir els cavalls de Troia és mitjançant la comparació dels fitxers, és a dir, es pot comparar un fitxer amb el fitxer original. Un sistema tradicional de fer això és un procés que emmagatzema la suma MD5 de tots els fitxers executables del sistema i, una vegada al dia, aquest procés compara la suma dels fitxers MD5 actuals del sistema amb els que es té emmagatzemat. Si el fitxer no té la mateixa suma, es notifica a l'administrador del sistema perquè analitzi el problema.

Bombes lògiques

Les bombes lògiques són part de codi d'un programa que a priori no realitza res, però que s'activa segons unes condicions o per part d'un tercer. Quan s'activa la bomba lògica, el programa farà accions per a les quals no ha estat creat; en general, aquestes bombes realitzen accions perjudicials i, fins i tot, obren *backdoors*. Activadors de les bombes lògiques poden ser dates, creació de fitxers, esborrament de registres d'una base de dades.

Backdoors

En els desenvolupaments grans és habitual per part dels programadors programar «dreceres» dins dels sistemes d'autenticació. Aquestes dreceres es coneixen com a *backdoors* o portes del darrere i s'utilitzen perquè, a l'hora de desenvolupar grans programes, ens puguem saltar autenticacions per anar més de pressa en la detecció d'errors i en la depuració del programa. El problema d'aquestes dreceres és quan després del desenvolupament no es tanquen, ja que un atacant pot descobrir aquestes portes i accedir als sistemes impunement. També pot ser utilitzat per personal que ja no treballa en l'empresa (exempleats) per accedir a dades sensibles.

A banda de portes del darrere en els programes, és possible –i típic– situar portes del darrere en certs fitxers vitals per al sistema; generalment, quan un atacant aconsegueix accés a una màquina del sistema, vol mantenir aquest accés encara que es detecti la seva penetració. Per exemple, una cosa molt habitual és afegir un usuari amb UID 0 en el fitxer de claus, de manera que el pirata pugui continuar accedint al sistema amb aquest nou *login* encara que l'administrador tanqui la porta que abans havia utilitzat per entrar. També és clàssic afegir un nou servei en un port no utilitzat, de manera que fent *telnet* a aquest número de port s'obri un *shell* amb privilegis d'administrador; fins i tot molts atacants utilitzen la facilitat *cron* per revisar periòdicament aquests arxius i inserir les portes del darrere de nou, en cas que hagin estat esborrades.

Què s'ha de fer per evitar aquests atacs? La prevenció passa per comprovar periòdicament la integritat dels arxius més importants (fitxers de contrasenyes, configuració de la xarxa, programes de l'arrencada de màquina...). També és convenient rastrejar l'existència de nous arxius amb privilegis d'execució d'administrador que puguin aparèixer en els sistemes de fitxers: qualsevol nou programa d'aquestes característiques sol indicar un atac d'èxit i una porta del darrere collocada en la nostra màquina.

Cros-site scripting (XSS)

Cros-site scripting o XSS és el nom que rep una vulnerabilitat que afecta no tant els servidors com els usuaris que naveguen en pàgines d'Internet. La causa de la vulnerabilitat radica en la pobra verificació per part dels llocs web de les cadenes d'entrada enviades pels usuaris a través de formularis, o directament, a través de

l'URL. Aquestes cadenes, en cas que siguin malicioses, podrien arribar a contenir *scripts* complets. Quan aquesta entrada es mostra dinàmicament a un usuari dins d'una pàgina web, en cas que contingui un *script*, aquest s'executarà en el navegador de l'usuari dins del context de seguretat de la pàgina web visitada. Com a conseqüència, podrà realitzar en l'ordinador de l'usuari totes les accions que li siguin permeses en aquest lloc web, com per exemple interceptar entrades de l'usuari víctima o llegir les seves *cookies*.

El risc més important d'aquest tipus d'atacs és que l'entrada maliciosa no la proporciona el mateix usuari que veu la pàgina, sinó un atacant, que aconsegueix que l'*script* s'executi en el navegador de l'usuari. La víctima executa el codi de manera indirecta quan confiadament fa clic sobre un hiperenllaç fraudulent, que pot ser present en el lloc web de l'atacant, en un missatge de correu electrònic o d'un grup de notícies, o en qualsevol altre lloc que no aixequi sospites. Com que en aquest cas la víctima és el visitant del lloc web i no el mateix lloc, aquest tipus de vulnerabilitats no ha rebut l'atenció que es mereix.

Per tant, el *cros-site scripting* funciona de la manera següent:

1. L'usuari segueix un enllaç, que inclou codificada una cadena d'entrada com a argument d'entrada a algun paràmetre de la pàgina del lloc web.
2. El lloc web no valida (o ho fa pobrament) l'entrada anterior i genera dinàmicament una pàgina HTML que inclou el codi introduït en l'hiperenllaç per l'atacant.
3. Aquest codi s'executa en el navegador de la víctima, amb els mateixos privilegis que qualsevol altre codi legítim del mateix lloc web.

3.7. Seguretat del sistema

Introducció

En les aplicacions web convé tenir en compte la seguretat del sistema que les contenen. La protecció dels sistemes no és fàcil, però els punts següents s'han de revisar en qualsevol sistema operatiu.

Nom d'usuari /contrasenya en blanc, per defecte o feble

No és estrany que els usuaris no canviïn les contrasenyes durant molt temps o siguin febles. Per evitar això, haurem de fer proves periòdiques de fortalesa de les contrasenyes, a més a més de posar-hi mesures restrictives, com per exemple forçar a una longitud mínima de 8 caràcters o contrasenyes que caduquen cada tres mesos.

Per provar la fortalesa de les contrasenyes podem utilitzar programes que per força bruta poden endevinar contrasenyes, com per exemple el «John the ripper» <http://www.openwall.com/john>.

Les polítiques de contrasenyes seran realitzades mitjançant les eines que ens indiqui el sistema operatiu.

Fitxers

Permisos dels fitxers

Haurem de modificar la màscara de creació dels fitxers perquè només els pugui modificar i visualitzar l'usuari rw-------. En Linux es correspon amb la màscara 0177 i s'estableix amb l'ordre *umask*. Només si és necessari, donarem permisos a nivell de grup.

Tindrem especial cura amb els permisos dels fitxers de configuració que continguin informació com contrasenyes. Per a aquests fitxers no ens quedarà cap més remei que construir-nos *scripts* personalitzats perquè ens «vigilin» els permisos d'aquests fitxers i, en cas que els permisos no siguin correctes, ens informin de l'error.

Suid i *Sgid*

Moltes vegades un procés necessita executar-se amb uns privilegis majors (o menors) que l'usuari que el va llançar. Per exemple, un usuari pot modificar la seva pròpia contrasenya amb l'ordre *passwd*, però això implica modificar el fitxer /etc/passwd, i per a això, un usuari normal no disposa d'aquest permís. Això se soluciona activant el bit SUID de l'ordre *passwd* (observeu que quan això passa, la *x* d'executable passa a ser una *s*):

```
ls -la /usr/bin/passw*
-r-sr-xr-x 1 root bin 15613 abr 27 1998 /usr/bin/passwd
```

Això vol dir que, quan s'executi, el procés corresponent tindrà els privilegis del propietari de l'ordre (és a dir, el *root*), no de l'usuari que el va llançar. En altres paraules, el procés generat per *passwd* pertany a *root*. Això és un problema de seguretat, encara que si el programa funciona correctament, no té per què donar problemes; però els *bugs* i els errors en els programes poden ser utilitzats per algú per intentar executar un altre codi diferent amb els privilegis d'aquest procés. El mètode sol donar problemes en els programes que hem indicat en l'apartat anterior.

Qualsevol atacant que hagi entrat en un sistema de manera il·legítima intentarà deixar una *shell* amb el bit SUID per mantenir aquest nivell de privilegi quan torni a entrar en el sistema.

SGID és el mateix que SUID, però aplicat al grup.

Així doncs, cal anar amb compte en els programes amb el bit SUID/SGIG. Es poden trobar amb

```
root# find / -type f \( -perm -04000 -o -perm -02000 \) -print
```

Cal tenir en compte que alguns programes (com *passwd*) han de tenir el bit SUID. Cal comprovar en els llocs habituals (que indiquem en la secció corresponent) que cap dels programes propietat del *root* o SUID que utilitzeu en el vostre sistema té un error de seguretat conegut que pugui ser explotat.

Serveis i servidors

Hem de tenir en compte els serveis i servidors que tenim en el nostre sistema funcionant. Només hem de tenir funcionant les que realment necessitem.

En un servidor web no té sentit que estigui funcionant un servidor d'impressió (*cups*) o un servidor *x*, per a això haurem de revisar els *runlevels* en els servidors Linux i només tenir funcionant les que necessitem.

Per revisar els serveis i servidors que tenim funcionant ho podem fer mitjançant el directori /etc/rc.d/ on trobarem els diferents *runlevels* o, per exemple, amb l'ordre «chkconfig».

```
servhost:~# chkconfig --list
acpid 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

```
atd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
avahi-daemon 0:off 1:off 2:on 3:on 4:on 5:on 6:off
bind9 0:off 1:off 2:on 3:on 4:on 5:on 6:off
bootlogd 0:off 1:off 2:off 3:off 4:off 5:off 6:off S:on
bootlogs 0:off 1:on 2:on 3:on 4:on 5:on 6:off
bootmisc.sh 0:off 1:off 2:off 3:off 4:off 5:off 6:off S:on
checkfs.sh 0:off 1:off 2:off 3:off 4:off 5:off 6:off S:on
checkroot.sh 0:off 1:off 2:off 3:off 4:off 5:off 6:off S:on
console-screen.sh 0:off 1:off 2:off 3:off 4:off 5:off 6:off S:on
cron 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Programes

És molt important controlar els programes que tenim instal·lats en els nostres sistemes. No hem de tenir cap programa instal·lat que no calgui per al funcionament de les nostres aplicacions. Per aquest motiu, haurem d'anar amb compte amb les instal·lacions per defecte.

D'altra banda, haurem de tenir actualitzades totes les aplicacions que tinguem instal·lades en el nostre sistema, amb la finalitat de no tenir problemes en les nostres aplicacions per errors en la programació d'aquestes. Per poder tenir identificades les actualitzacions de seguretat dels programes, podem estar subscrits a les notícies de les nostres distribucions GNU/Linux, però també podem visitar pàgines web on es publiquen totes les vulnerabilitats detectades, o subscriure'ns-hi. La més típica és Common Vulnerabilities and Exposures <http://cve.mitre.org>. A Debian. per exemple, podem seguir el compte de Twitter on publiquen les actualitzacions de seguretat <@debiansecurity>.

Revisions dels fitxers de *log*

Una altra de les rutines que hem de tenir és revisar els fitxers de *log* del sistema i de les aplicacions.

Per fer això, podem analitzar els *logs* amb aplicacions com Logwatch, que ens permeten automatitzar l'anàlisi dels *logs* del sistema.

Els *logs* més importants que hauríem de revisar diàriament són:

- *syslog*: normalment s'ubica en /var/log/syslog, i hi podem trobar tots els missatges del sistema. És important comprovar que no tenim cap error de cap tipus, sigui de maquinari, de processos programats, etc.

```
servhost:/var/log# tail -10 /var/log/syslog
Oct 30 23:45:01 servhost /USR/SBIN/CRON[18893]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
Oct 30 23:55:01 servhost /USR/SBIN/CRON[19431]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
Oct 30 23:58:16 servhost dhcpd: DHCPREQUEST for 192.168.1.181 from 52:54:00:25:66:db via br0
Oct 30 23:58:16 servhost dhcpd: DHCPACK on 192.168.1.181 to 52:54:00:25:66:db via br0
Oct 30 23:59:01 servhost /USR/SBIN/CRON[19702]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 60 2)
Oct 31 00:05:01 servhost /USR/SBIN/CRON[20241]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
Oct 31 00:15:01 servhost /USR/SBIN/CRON[20781]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
Oct 31 00:17:01 servhost /USR/SBIN/CRON[21052]: (root) CMD ( cd / && run-parts --report /etc/cron.hourly)
Oct 31 00:25:01 servhost /USR/SBIN/CRON[21599]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
Oct 31 00:35:01 servhost /USR/SBIN/CRON[22157]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
servhost:/var/log#
```

- *auth.log* : normalment s'ubica en /var/log/auth.log, on trobarem tots els registres d'intents d'entrades al sistema, des dels que han tingut èxit fins als que han estat fallits. També tindrem els accessos de canvis d'usuari mitjançant l'eina *su*.

```
servhost:/var/log# tail -10 /var/log/auth.log
Oct 31 00:17:01 servhost CRON[21050]: pam_unix(cron:session): session opened for user root by (uid=0)
Oct 31 00:17:01 servhost CRON[21050]: pam_unix(cron:session): session closed for user root
Oct 31 00:25:01 servhost CRON[21597]: pam_unix(cron:session): session opened for user root by (uid=0)
Oct 31 00:25:01 servhost CRON[21597]: pam_unix(cron:session): session closed for user root
```

```
Oct 31 00:33:17 servhost su[22137]: Successful su for root by root
Oct 31 00:33:17 servhost su[22137]: + /dev/pts/0 root:root
Oct 31 00:33:17 servhost su[22137]: pam_unix(su:session): session opened for user root by root(uid=0)
Oct 31 00:33:20 servhost su[22137]: pam_unix(su:session): session closed for user root
Oct 31 00:35:01 servhost CRON[22155]: pam_unix(cron:session): session opened for user root by (uid=0)
Oct 31 00:35:01 servhost CRON[22155]: pam_unix(cron:session): session closed for user root
servhost:/var/log#
```

- *Messages*: normalment ubicada en /var/log/messeges aquí s'emmagatzemen dades «informatives» de certs programes, missatges de baixa o mitjana prioritat destinats a informar i a avisar de successos importants, com informació relativa a l'arrencada de la màquina o a serveis del sistema:

```
servhost:/var/log# tail -10 /var/log/messages
Oct 30 21:03:17 servhost dhcpd: DHCPACK on 192.168.1.181 to 52:54:00:25:66:db via br0
Oct 30 22:01:22 servhost dhcpd: DHCPREQUEST for 192.168.1.181 from 52:54:00:25:66:db via br0
Oct 30 22:01:22 servhost dhcpd: DHCPACK on 192.168.1.181 to 52:54:00:25:66:db via br0
Oct 30 22:59:17 servhost dhcpd: Wrote 0 deleted host decls to leases file.
Oct 30 22:59:17 servhost dhcpd: Wrote 0 new dynamic host decls to leases file.
Oct 30 22:59:17 servhost dhcpd: Wrote 13 leases to leases file.
Oct 30 22:59:17 servhost dhcpd: DHCPREQUEST for 192.168.1.181 from 52:54:00:25:66:db via br0
Oct 30 22:59:17 servhost dhcpd: DHCPACK on 192.168.1.181 to 52:54:00:25:66:db via br0
Oct 30 23:58:16 servhost dhcpd: DHCPREQUEST for 192.168.1.181 from 52:54:00:25:66:db via br0
Oct 30 23:58:16 servhost dhcpd: DHCPACK on 192.168.1.181 to 52:54:00:25:66:db via br0
servhost:/var/log#
```

- *wtmp* : Aquest arxiu és un fitxer binari que emmagatzema informació relativa a cada connexió i desconnexió del sistema. S'ha de revisar amb l'ordre *last*.

```
servhost:/var/log# last -10
root pts/0 ptmmm.local Sun Oct 30 20:06 still logged in
marco pts/0 83.247.136.133 Tue Oct 4 16:40 - 18:41 (02:00)
wtmp begins Tue Oct 4 16:40:28 (02:00)
```

3.8. Seguretat en bases de dades

Introducció

En les aplicacions web és convenient protegir les bases de dades que s'utilitzen des de les aplicacions web. La protecció de bases de dades no és trivial, ja que de vegades s'aprofiten les vulnerabilitats més senzilles, com per exemple contrasenyes febles.

Errors de seguretat en base de dades

Tot i que poden ser errors que ja s'han presentat en apartats anteriors, oferim una guia per tenir en compte què hem de revisar en una base de dades.

Nom d'usuari /contrasenya en blanc, per defecte o feble

No és gens estrany aconseguir el dia a dia parells d'usuari/password com sa/1234. Aquesta és la primera línia de defensa i un punt fonamental de l'armadura de les nostres bases de dades. És important fer revisions periòdiques de credencials.

Injeccions SQL

Quan la plataforma de base de dades falla per desinfectar les entrades, els atacants són capaços d'executar les injeccions SQL de manera similar a com ho fan en els atacs basats en web, i això els permet elevar els seus privilegis i obtenir accés a una àmplia gamma de funcionalitats. Molts dels proveïdors han donat a conèixer solucions per solucionar aquests problemes en els programes de base de dades, però no servirà de gaire si els pedaços no s'apliquen o no s'adopten els correctius corresponents.

Preferència de privilegis d'usuari respecte de privilegis de grup

Les empreses necessiten garantir que els privilegis no es donin als usuaris per assignació directa. Es recomana que els usuaris només rebin privilegis per part de grups o funcions i siguin manejats col·lectivament. D'aquesta manra, serà més fàcil eliminar drets a un usuari simplement eliminant-lo del grup, sense que quedin drets ocults o oblidats assignats a l'usuari esmentat.

Característiques de base de dades innecessàriament habilitades

Cada instal·lació de base de dades ve amb paquets addicionals de totes les formes i mides que, en la seva majoria, rarament són utilitzats per una sola organització. Atès que hi ha la possibilitat de tenir programes amb errors, cal veure els paquets que no són necessaris i desinstal·lar-los o no instal·lar-los en la instal·lació inicial. Això no solament redueix els riscos d'atacs, sinó que també simplifica la gestió de pedaços.

Configuració de seguretat ineficient

De la mateixa manera, les bases de dades tenen una gran quantitat d'opcions de configuració i consideracions diferents de disposició dels administradors per ajustar el rendiment i les funcionalitats millorades. Les organitzacions necessiten aconseguir i desactivar les configuracions insegures que podrien estar activades per defecte per a més comoditat dels DBA o desenvolupadors d'aplicacions. Les

configuracions de bases de dades en producció i desenvolupament han de ser radicalment diferents.

Buffer overflow

Les vulnerabilitats de *buffer overflow* són explotades per les entrades amb valors diferents o molt superiors als que l'aplicació espera –per exemple, mitjançant l'addició de 100 caràcters en un quadre d'entrada demanant un nombre de la Seguretat Social. Els proveïdors de bases de dades han treballat dur per solucionar els problemes tècnics que permeten que aquests atacs es produeixin. Aquesta és una altra raó per la qual els pedaços són tan importants.

Escalada de privilegis

De la mateixa manera, les bases de dades sovint exposen vulnerabilitats comunes que permeten a un atacant escalar privilegis en un compte de privilegis baixos fins a tenir accés als drets d'un administrador. A mesura que aquestes vulnerabilitats són descobertes, els proveïdors les corregeixen i els administradors han de mantenir les actualitzacions i els pedaços actualitzats.

Atac de denegació de servei

El cas de l'SQL Slammer és sempre un exemple molt aclaridor de com els atacants poden utilitzar les vulnerabilitats dels DBMS per enderrocar els servidors de base de dades a través d'un alt flux de trànsit. Encara més il·lustratiu és el fet que, quan l'Slammer va atacar el 2003 un pedaç que ja estava per aquí es va dirigir a corregir la vulnerabilitat per la qual es va generar el seu atac. Avui dia, set anys més tard, SQL Slammer encara està donant mals de cap en els servidors no actualitzats.

Bases de dades sense actualitzar

Això podria sonar repetitiu, però val la pena repetir-ho. Els administradors de base de dades de vegades no apliquen un pedaç en el moment oportú, perquè tenen

por que aquest danyi les seves bases de dades. Però el risc de patir l'atac d'un *hacker* avui és molt més alt que el risc d'aplicar un pedaç que descompongui la base de dades. A més a més, davant aquests temors hi ha les còpies de seguretat i les rèpliques.

3.9. Dades sense xifrar

Potser sigui una obvietat, però les organitzacions no han d'emmagatzemar les dades sensibles en text pla en una taula. I totes les connexions a la base de dades sempre que manegin dades sensibles han d'utilitzar el xifratge.

Auditoria de seguretat

Introducció

En els apartats anteriors hem presentat els diferents punts relacionats amb els problemes de seguretat i com evitar-los.

Aquestes auditories de seguretat s'haurien de fer com a mínim un cop l'any, encara que el més aconsellable és que es facin cada mes o cada tres mesos.

En els punts següents presentarem com podem posar els nostres sistemes a prova i quines mesures que podem prendre per comprovar la integritat dels nostres sistemes.

Contrasenyes per defecte o febles

Això ja ho hem presentat en apartats anteriors, però bàsicament el que hem de fer és provar les contrasenyes que tenen els nostres usuaris. Per fer aquesta funció utilitzarem programes com el John the ripper.

SUID i SGID

Revisarem que els fitxers que contenen el *suid* i *sgid* no han canviat. Per a això podem emmagatzemar amb l'ordre següent:

```
root# find / -type f \( -perm -04000 -o -perm -02000 \) -print
```

i emmagatzemar la seva sortida en un fitxer que comprovarem periòdicament amb la finalitat de veure si hi ha algun canvi en els fitxers que contenen aquests permisos.

Programes d'auditoria

En els apartats següents presentarem eines que ens permetran analitzar una o moltes vulnerabilitats dels nostres sistemes.

Nessus

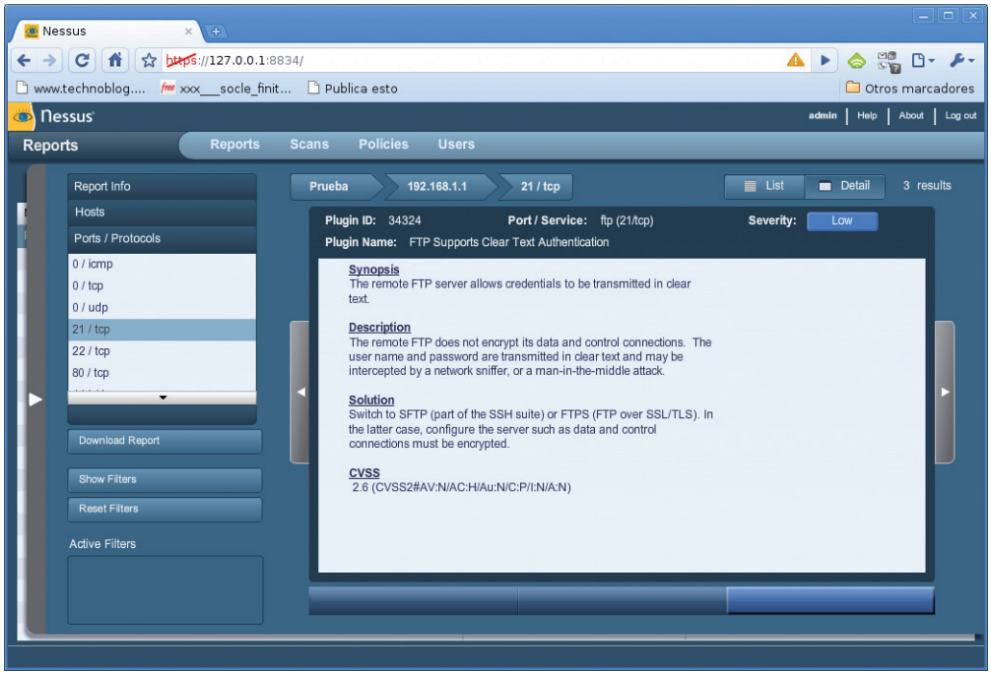
Nessus és un programa d'escaneig de vulnerabilitats en diversos sistemes operatius. Es compon de Daemon Nessus, que realitza l'escaneig en el sistema objectiu, i de Nessus, el client (basat en consola o gràfic), que mostra l'avanç i report dels escanejos. Des de la consola, Nessus pot ser programat per fer escanejos programats amb *cron*.

En operació normal, Nessus comença escanejant els ports amb nmap o amb el seu propi escanejador de ports per buscar ports oberts i després intentar diversos *exploits* per atacar-lo. Les proves de vulnerabilitat, disponibles com una llarga llista de *plugins*, estan escrites en NASL (Nessus Attack Scripting Language, és a dir, llenguatge de *scripting* d'Atac Nessus), un llenguatge *scripting* optimitzat per a interaccions personalitzades en xarxes.

Opcionalment, els resultats de l'escaneig es poden exportar en reports en diversos formats, com text pla, XML, HTML i LaTeX. Els resultats també es poden desar en una base de coneixement per a referència en futurs escanejos de vulnerabilitats.

Algunes de les proves de vulnerabilitats de Nessus poden causar que els serveis o sistemes operatius es corrompin i caiguin. L'usuari pot evitar això desactivant «unsafe test» (proves no segures) abans d'escanejar.

El funcionament típic és accedir a la consola del programa Nessus i llançar l'anàlisi sobre una IP.



Pantalla de detecció de possible vulnerabilitat en Nessus

Tripwire

L'eina Tripwire és un comprovador d'integritat per a fitxers i directoris de sistemes Unix: compara un conjunt d'aquests objectes amb la informació sobre aquests emmagatzemada prèviament en una base de dades i alerta l'administrador en cas que alguna cosa hagi canviat. La idea és simple: es crea un resum de cada fitxer o directori important per a la nostra seguretat només instal·lar el sistema i aquests resums s'emmagatzemen en un mitjà segur (un CD-ROM o un disc protegit contra escriptura), de manera que si algun dels fitxers és modificat (per exemple, per un atacant que substitueix un programa per una versió troiana, o afegeix una entrada en el nostre fitxer de contrasenyes) Tripwire ens alertarà la pròxima vegada que fem la comprovació. Per generar aquests resums, s'utilitzen funcions *hash*, de manera que gairebé és impossible que dos fitxers generin el mateix resum; concretament Tripwire implementa MD2, MD4, MD5, Snefru, CRC-16 i CRC-32.

Nmap

Nmap és una eina d'escaneig de ports per a grans xarxes. Pot escanejar des de màquines individuals fins a segments grans de xarxa. Permet diversos models d'escaneig, segons les proteccions que tingui el sistema. També té tècniques per determinar el sistema operatiu que utilitzen, etc.

En l'exemple següent podem veure quins ports hi ha oberts i la raó per la qual detecta els oberts (syn-acl):

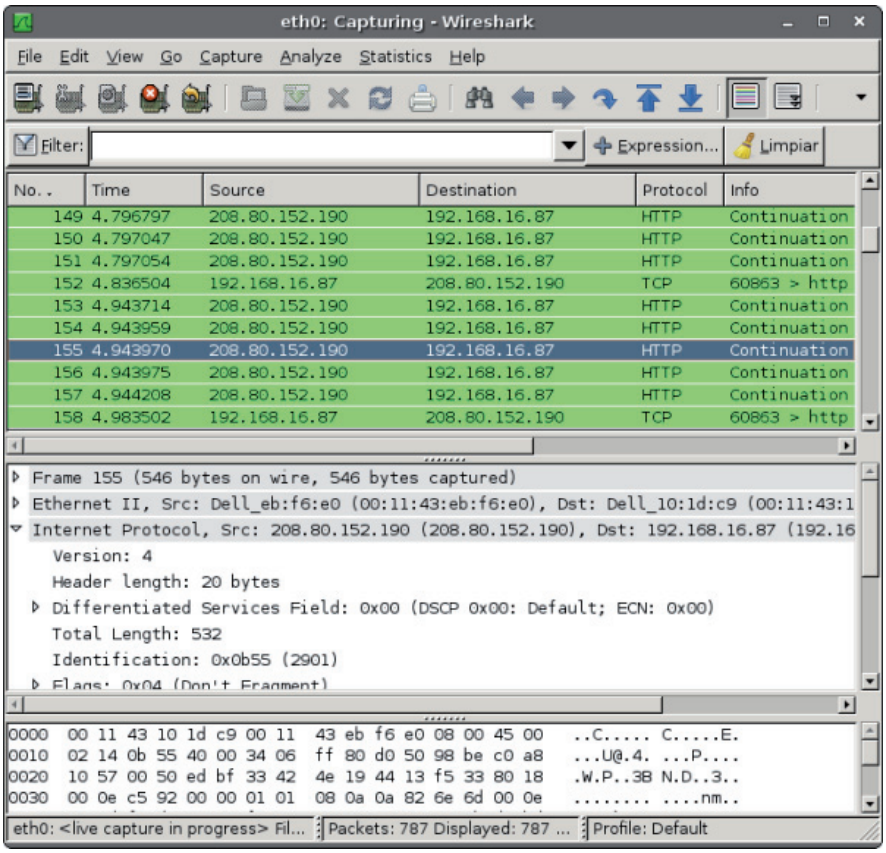
```
# nmap 192.168.1.2 -sS --reason

Starting Nmap 5.00 ( http://nmap.org ) at 2011-10-31 01:48 CET
Interesting ports on 192.168.1.2:
Not shown: 993 closed ports
Reason: 993 resets
PORT STATE SERVICE REASON
22/tcp open ssh syn-ack
53/tcp open domain syn-ack
111/tcp open rpcbind syn-ack
139/tcp open netbios-ssn syn-ack
445/tcp open microsoft-ds syn-ack
631/tcp open ipp syn-ack
49152/tcp open unknown syn-ack

Nmap done: 1 IP address (1 host up) scanned in 0.34 seconds
#
```

Wireshark

Wireshark és un capturador/analitzador de paquets de xarxa (anomenat de vegades sniffer). Wireshark ens permetrà veure, a un nivell baix i detallat, el que està passant en la xarxa. A més a més és gratuït, *open source* i multiplataforma. És una de les millors eines a l'hora d'auditar la xarxa.



Exemple de captura de Whireshark

Snort

Snort és un IDS o sistema de detecció d'intrusions basat en xarxa (NIDS). Implementa un motor de detecció d'atacs i escombratge de ports que permet registrar, alertar i respondre davant qualsevol anomalia prèviament definida com patrons que corresponen a atacs, escombratges, intents d'aprofitar alguna vulnerabilitat, anàlisi de protocols, etc., Tot això en temps real.

Snort <http://www.snort.org/> està disponible sota llicència GPL, és gratuït i funciona en plataformes Windows i UNIX/Linux. És un dels més usats i disposa d'una gran quantitat de filtres o patrons ja predefinits, i d'actualitzacions constants davant casos d'atacs, escombratges o vulnerabilitats que es detectin a través dels diferents butlletins de seguretat.

3.10. Còpies de seguretat

S'exposaran les solucions i els mecanismes de còpia de seguretat. Es parla del concepte de «pla de còpies de seguretat».

En els actuals sistemes d'emmagatzematge poden fallar i produir d'aquesta manera una pèrdua de dades parcial o total. Per a aquestes situacions hem de tenir unes còpies de seguretat o suport.

La còpia de seguretat o còpia de suport es refereix a una còpia que, en cas de pèrdua d'informació, es pugui restaurar a un punt anterior de l'estat del sistema. La còpia de seguretat és útil per diverses raons, com per exemple per restaurar un sistema al seu estat anterior a un desastre o per restaurar un fitxer o uns fitxers després d'una eliminació accidental.

Dit d'una altra manera, la còpia de seguretat és la còpia total o parcial d'informació important del disc dur, bases de dades o un altre mitjà d'emmagatzematge. Aquesta còpia de seguretat s'ha de fer en un altre sistema d'emmagatzematge massiu, com ara discos durs, CD, DVD o cintes magnètiques.

Motivacions de les còpies de seguretat

Totes les dades amb les quals treballem diàriament en un PC, el servidor les desa en un mitjà d'emmagatzematge, i això implica que la supervivència de tot treball realitzat depèn del funcionament correcte d'aquests sistemes. De vegades s'espatllen parts d'aquests sistemes que no produeixen pèrdua d'informació (per exemple, un ratolí, un teclat, una pantalla) que es poden substituir, però altres produeixen pèrdues d'informació.

De vegades hi ha problemes amb els programes. Errors humans que poden produir pèrdua de dades.

Alguns dels motius que provoquen més pèrdua de dades els detallem a continuació:

- Fallades de dispositius d'emmagatzematge. Els dispositius d'emmagatzematge estan formats per diversos components electrònics i mecànics. Aquests components poden fallar. Aquestes fallades poden ser ocasionades per defectes de fabricació o d'un altre tipus, fins i tot per tercers, i poden produir la pèrdua parcial o total de les dades.
- Error humà. Intencionat o no a, vegades es produeixen pèrdues de dades a causa d'operacions que realitzen els usuaris sobre les dades.

- Operacions incorrectes d'un programa. Poden ser intencionades o no, però mitjançant l'operació incorrecta d'una aplicació es poden eliminar arxius que continguin dades.
- Atacs externs. Els virus o els ingressos d'usuaris sense control poden causar greus pèrdues d'informació.
- Desaparició del dispositiu d'emmagatzematge. A part dels problemes anteriors, s'ha de tenir en compte que es pot produir una pèrdua del dispositiu d'emmagatzematge, ja sigui per sinistre, robatori o per un desastre natural.

Per tots aquests motius i alguns altres cal tenir còpies de seguretat per assegurar-se la continuïtat del negoci davant qualsevol desastre.

Per exemple, podem imaginar el que pot passar si un banc perd totes les dades dels seus clients o si una botiga web perd les dades de les seves comandes i tot el desenvolupament web resulta eliminat.

Què hem d'exigir a una còpia de seguretat?

Explicarem què hem d'exigir a una còpia de seguretat, què ha de tenir i quines característiques ha de complir.

Atesa la importància de les còpies de seguretat dins els plans de seguretat d'una organització, és normal que aquestes hagin de complir uns requisits mínims. Podem enumerar els següents, que anirem explicant en els apartats següents:

- Posseir una planificació de les còpies. En les planificacions de les còpies de seguretat no sempre es fa una còpia del mateix; per aquest motiu s'han de tenir en compte quines polítiques de còpies i de rotació de dispositius tenim implementades. Aquest requisit ens obliga a tenir documentada tota la nostra planificació de còpies de seguretat, per tal de saber què, quan i on s'ha fet la còpia de seguretat d'una dada determinada.
- Tenir consistència i fiabilitat en les còpies de seguretat. Per complir aquest requisit, hem de tenir en compte dues coses. D'una banda, la integritat de les dades desades, és a dir, que no estiguem copiant dades corruptes. De l'altra, assegurar-nos que els dispositius i els suports on estem fent les còpies de seguretat estan en bon estat.

- Seguretat d'emmagatzematge. És convenient que els suports on es fan les còpies de seguretat estiguin emmagatzemats en un lloc segur. Quan ens referim a seguretat, hem d'aclarir que és tant accés restringit a persones com contra desastres naturals, com per exemple un incendi o una inundació.
- Recuperació ràpida i fiable de la informació. Una vegada realitzada la còpia de seguretat, hem d'estar segurs que podrem fer una recuperació de qualsevol informació que estigui en la còpia de seguretat en el menor temps possible (això podrien ser dies, en cas de quantitats ingents d'informació; en tot cas, ha de ser el temps més breu possible). També hem d'estar segurs que aquesta informació recuperada ha de ser el que volem recuperar.
- Desaparició del dispositiu d'emmagatzematge. A part dels problemes anteriors, s'ha de tenir en compte que es pot produir una pèrdua del dispositiu d'emmagatzematge, sigui per sinistre, robatori o per un desastre natural.

Conceptes tècnics

Explicarem alguns dels conceptes tècnics bàsics abans d'entrar en els apartats següents en què aniran apareixent.

El sistema de còpies de seguretat és senzill. Normalment es té una informació origen en un dispositiu d'emmagatzematge que, mitjançant un programa dissenyat per a còpies de seguretat, es còpia en un dispositiu d'emmagatzematge destí (per exemple, cintes, discos).

L'origen de les dades pot ser variat, des d'una carpeta o directori, fins a una còpia completa d'un servidor. L'important i el que hem de tenir clar és el que es còpia, que són els continguts lògics.

Si es realitzen còpies de seguretat de molta informació, s'han d'utilitzar diversos volums o cintes, que anomenarem conjunt de còpies de seguretat.

Els programes de còpia de seguretat, a part d'encarregar-se de copiar la informació que nosaltres volem, desen un índex de tot el que s'ha copiat. Aquest índex normalment l'anomenem catàleg. Aquest catàleg es genera quan es realitzen les còpies de seguretat, i deixen constància del que s'ha copiat i de la data de la còpia. Amb aquest catàleg podrem recuperar un fitxer determinat en diferents dates en cas que es produeixi qualsevol problema en aquest fitxer (esborrament, modificació errònia, etc.).

Anomenem sessió de còpia de seguretat l'especificació del que volem copiar en cada sessió de còpia de seguretat.

Tipus de còpies de seguretat

En aquest apartat explicarem els diferents tipus de còpies de seguretat a l'hora de fer una còpia de seguretat.

Els tipus de còpies de seguretat es poden classificar, a grans trets, en dos tipus:

- Còpies de tot el que hi ha en l'origen: còpies completes.
- Còpies de tot el que s'ha modificat: còpies incrementals i còpies diferencials.

Còpies completes

Les còpies completes són les que realitzen una còpia de seguretat de «tot». Aquest tipus de còpia de seguretat còpia tot el que s'especifica en l'origen i ho emmagatzema en destí. El destí poden ser diferents tipus de suports.

Una cosa que hem de tenir en compte quan fem aquest tipus de còpies és que solen ocupar molt espai i requereixen molt temps per ser fetes.

No obstant això, són les que permeten una restauració de la informació completa d'una manera més fàcil i ràpida. Només haurem de triar la còpia completa del dia que volem la restauració i on restaurar.

Quan diem que ho copien tot, és literal: es còpia tot el que està especificat en la sessió de còpia de seguretat, sense tenir en compte canvis ni quins fitxers han estat modificats. Això permet que, si tenim una sessió de còpia de seguretat d'un servidor al complet, en cas d'error del maquinari del servidor, podrem restaurar el servidor al complet en un altre servidor que tingui el mateix maquinari sense cap problema.

Amb el que hem dit fins ara, aquestes còpies s'haurien de fer abans de canvis importants en els sistemes. O determinades còpies s'han de desar durant molt temps (còpies anuals, mensuals).

Normalment són les còpies que es realitzen la primera vegada en qualsevol política de suport.

Còpies incrementals

Les còpies incrementals són les còpies de seguretat que es realitzen dels fitxers nous o modificats després de l'última còpia de seguretat, sigui incremental o completa.

També s'anomenen còpies evolutives. Cada vegada que es realitza una còpia d'aquest estil, no es desen tots els fitxers, només es desen els fitxers creats o modificats. Això es du a terme utilitzant el bit de modificació.

Per fer aquest tipus de còpies el programa de còpia de seguretat s'encarrega de revisar els fitxers que s'han modificat i els fitxers creats mitjançant el bit de modificació i posteriorment els copia en el dispositiu de còpia. També s'encarrega de generar el catàleg de fitxers copiats aquest dia. Una vegada copiat el fitxer, es desmarca el bit de modificació (i els marca com a copiats).

El benefici d'aquestes còpies està en el fet que es copia menys informació i s'utilitza menys temps en la còpia de seguretat. També es produeix un estalvi en la utilització dels dispositius de còpia, ja que, com que es copia menys informació, s'utilitzen menys recursos.

Alguns dels inconvenients de les còpies incrementals són que depenem del catàleg del programa de còpia de seguretat per poder identificar i restaurar la informació; per tant, hem de tenir cura d'administrar-la correctament.

D'altra banda, les recuperacions totals d'informació són més complexes, ja que hem de partir de l'última còpia completa que s'ha fet i anar restaurant còpies incrementals de mica en mica fins a arribar al dia d'avui. Això pot ser molt molest, ja que poden estar implicats un bon nombre de dispositius de còpia. Tanmateix, es pot solucionar si tenim alguns dispositius, com les sitges de cintes o els robots de còpies que posen i treuen automàticament les cintes per a una recuperació més ràpida.

De qualsevol manera, la utilització de còpies incrementals no ens evita de fer cada cert temps còpies completes.

Còpies diferencials

Les còpies diferencials són les còpies de seguretat que es realitzen dels fitxers modificats des de l'última còpia completa o diferencial. Des d'aquest punt de vista, són com les còpies incrementals, però, a diferència d'aquestes, no utilitzen el bit de modificació, utilitzen la data de modificació.

El funcionament és el següent: el programa de còpies de seguretat examina els fitxers per la seva data de modificació i desa tots els fitxers des de l'última còpia

completa. És un tipus de còpia de seguretat similar a l'incremental. La diferència és que, com que no modifica el bit de modificació, còpia un fitxer que s'ha modificat en totes les sessions de còpia de seguretat fins que es realitza la pròxima còpia de seguretat completa.

Aquest tipus de còpia de seguretat utilitza menys espai que una còpia completa, però més que una d'incremental, ja que va copiant cada vegada els mateixos fitxers modificats fins a la pròxima còpia completa. Un problema pot estar en els sistemes amb molts canvis en tots els fitxers. Pot provocar que cada vegada que es facin còpies diferencials es copiïn tots els fitxers i s'acostin a la mida de la còpia completa. Per aquest motiu és molt important buscar la seqüència correcta de còpies completes i diferencials.

D'altra banda, la recuperació és molt més ràpida que en les còpies incrementals, ja que només necessitem restaurar l'última còpia completa i l'última còpia diferencial.

En la taula resum següent podem veure les característiques de cada mètode de còpia de seguretat.

Suport	Arxius en suport	Arxiu bit	Avantatges	Desavantatges
Complet	Tots	S'elimina de tots els arxius	Conté tota la informació	Temps en la realització de còpies.
Incremental	Arxius amb arxiu bit actiu.	Eliminat dels arxius dels quals es fa còpia de seguretat	Velocitat	Requereix l'última còpia completa i totes les còpies incrementals fins al dia que es vol fer la restauració completa.
Diferencial	Arxius amb arxiu bit actiu.	No s'elimina.	Només requereix l'última còpia completa i l'última còpia diferencial	Ocupa més espai que la còpia incremental

Taula comparativa de tipus de còpies

Mètodes de còpies de seguretat

A l'hora de fer còpies de seguretat es poden utilitzar diferents metodologies de còpia.

Normalment, llançarem una sessió de còpia de seguretat sobre un volum des del programa de còpies. Normalment, farem una còpia «en línia», ja que és difícil fer còpies en fred de volums.

En els apartats següents presentarem les diferents metodologies que podem utilitzar a l'hora de fer una còpia de seguretat.

Còpies en fred

És una metodologia difícil de realitzar, ja que normalment no podem desactivar un volum del seu ús per fer una còpia sense que ningú estigui accedint a les dades. A causa d'això s'utilitzen molt poques vegades o s'utilitzen en conjunció amb altres, com els *snapshots*, que veurem més endavant, que realitzen una parada en l'accés al volum de durada molt curta i que permeten posteriorment fer una còpia en fred.

Còpies en línia

Les còpies en línia són les còpies de seguretat que es realitzen sobre fitxers, bases de dades o dispositius en ús. Aquestes còpies de seguretat responen a un tipus de preguntes com el següent: qui pot tenir una aplicació aturada que té milions de fitxers per copiar i que triga 12 hores a fer la còpia de seguretat?

Aquestes còpies de seguretat es realitzen en entorns on no es poden «aturar» i que sempre han d'estar disponibles. També es realitzen si s'han de copiar quantitats ingents d'informació i la còpia de seguretat pot tardar molt temps.

Aquest tipus de còpies de seguretat tenen uns problemes detectats, que són els següents:

- Problemes si es realitzen canvis en els directoris/carpetes mentre es realitza la còpia de seguretat.
- Problemes si es modifiquen fitxers durant la còpia de seguretat.

Aquests problemes se solucionen amb diverses estratègies dins dels programes de còpies de seguretat, com per exemple bloquejar un fitxer mentre s'està copiant

o, en detectar que un fitxer es modificarà, copiar aquest fitxer abans que es modifiqui en el sistema de fitxers.

En el cas de bases de dades, aquests problemes se solucionen de diverses maneres. Algunes bases de dades poden treballar en mode arxivament (són un conjunt de fitxers que emmagatzemen les operacions que s'estan realitzant) i mentre es fa la còpia de seguretat de la base de dades s'emmagatzemen les operacions en els fitxers d'arxivament, i posteriorment també es fa la còpia d'aquests fitxers per tenir una còpia de seguretat consistent de la base de dades.

Snapshots

Els *snapshots* són una còpia de seguretat que consisteix en la còpia instantània d'un volum. L'*snapshot* és un volum de només lectura separat, que roman inalterable des del moment en què es crea. Aquest pot ser d'igual o d'una mida menor que el volum original, això a causa del fet que l'*snapshot* conté apuntadors al volum original i, quan es produeix un canvi en el volum original, aquest canvi es copia en el volum de l'*snapshot*. Per això podem tenir *snapshots* més petits que els volums originals. Tenint en compte això, haurem d'anar amb compte amb volums en els quals es produeixen molts canvis en tots els seus fitxers, ja que necessitarem la mateix mida que el volum original.

Fer un *snapshot* ens servirà per realitzar una còpia de seguretat (d'aquest) de manera normal sense els problemes que es poden produir en fer còpies en línia.

Còpies concurrents

En si no és un mètode de còpies de seguretat, però és una característica que hauria de tenir el programa de còpia. Bàsicament, consisteix en el fet que el programari de còpia de seguretat tingui la capacitat fer simultàniament la còpia de múltiples sistemes de fitxers sobre múltiples dispositius de còpia. Aquesta característica ens permetrà fer concurrentement diferents sessions de *backup* i reduir el temps en què es fan les còpies de seguretat.

Aquest tipus de còpies ens permetran la utilització de múltiples dispositius de còpia. En entorns molt grans (entorns amb centenars de servidors), ens permetran acabar les còpies de seguretat de tots els sistemes en un temps raonable (hores).

Suports d'emmagatzematge

En aquest apartat explicarem els diferents tipus de dispositius d'emmagatzematge per a les còpies de seguretat. Entre aquests, exposarem els dos dispositius més importants a l'hora de fer còpies de seguretat:

- Cintes
- Discos

Cintes

Les cintes són dispositius similars a les antigues cintes d'àudio, en què la informació s'emmagatzema en una «cinta» magnètica. Dins de les cintes, hi ha diversos tipus de cintes de còpia, com ara DDS, DLT, LTO i altres. D'aquests tipus, els principals són les cintes DDS i les LTO, que comentarem a continuació:

- Cintes DDS/DAT: deriven directament de les antigues cintes DAT (Digital Audio Tape) d'àudio. Tenen les generacions següents, que podem resumir en la taula següent:

Generació	Any	Capacitat (GB)	Capacitat comprimida (GB)	Velocitat (MB/s)
DDS-1	1989	1,3/2.0	2,6/4	0.18
DDS-2	1993	4,0	8	0.6
DDS-3	1996	12,0	24	1.1
DDS-4	1999	20,0	40	3,2
DAT-72	2003	36,0	72	3,2
DAT-160	2007	80	160	6,9
DAT-320	2009	160	320	12

Taula amb les característiques de les cintes DDS

Actualment, és una tecnologia obsoleta, però es continua utilitzant, ja que hi ha fabricants que continuen venent dispositius de còpia a baix preu.

DDS

- Cintes LTO/Ultrium: LTO ve de Linear Tape-Open i és un format obert que van desenvolupar conjuntament IBM, HP i Seagate en contrapartida del format DLT de l'empresa Quantum, que finalment ha estat desbancat. Aquestes cintes tenen unes grans capacitats i velocitats d'escriptura alta. En la taula següent podem veure l'evolució de la família de cintes LTO:

Generació	Any	Capacitat (GB)	Capacitat comprimida (GB)	Velocitat (MB/s)
LTO-1	2000	100	200	20
LTO-2	2003	200	400	40
LTO-3	2005	400	800	80
LTO-4	2007	800	1600	120
LTO-5	2010	1500	3000	140

Taula amb les característiques de les cintes LTO

Com podem comprovar en la taula, aquesta tecnologia és superior a les cintes DDS. Actualment és la tecnologia de cinta més evolucionada i amb més futur.

LTO

L'emmagatzematge de còpies de seguretat en cinta és tradicionalment més important, ja que el preu per MB era més baix que altres dispositius de còpia de seguretat. Actualment, els sistemes de còpia de cinta no es limiten a un dispositiu de còpia, hi ha complexos sistemes on s'emmagatzemen cintes i es carreguen automàticament en la unitat de cinta a mesura que la còpia de seguretat es va fent i el programa de còpies de seguretat va demanant diferents cintes on emmagatzemar les còpies que va fent. Aquests dispositius s'anomenen robots de cintes o llibreries.

Librería ORACLE

Tot i ser el principal sistema de còpies de seguretat, presenten alguns inconvenients que enumerem tot seguit:

- La informació no es troba disponible en qualsevol moment. S'ha de procedir a recuperar la informació.
- Les velocitats de còpia són inferiors als discos, encara que va en augment.
- Tenen una vida finita, encara que els dispositius/cintes LTO són molt més fiables que els dispositius/cintes DDS.

Com a avantatge podem destacar el següent:

- Són les que tenen una millor relació preu/GB.
- Són més fàcils de desar i traslladar (se'n comenta la importància en l'apartat d'on desar la còpia).
- En cas d'error d'una cinta, es pot substituir per una altra.

Discos

A causa de la baixada de preus, s'han començat a utilitzar discos durs per realitzar còpies de seguretat. La còpia sobre discos durs ha evolucionat d'una simple còpia sobre un disc als nous dispositius de còpia basats en discos.

Podem trobar dispositius de còpies que utilitzen unitats de disc dur en lloc d'unitats de cintes.

Aquests nous dispositius tenen els avantatges següents:

Sistema de copias de disco de HP

- Tenen ràtios de còpia més ràpids que les cintes.
- Permeten un accés molt més ràpid a la informació.
- Permet fer còpies que amb les unitats de cinta no es poden fer mitjançant *snapshots*.

I els inconvenients següents:

- En cas d'error, podem perdre tota la informació de totes les còpies.
- Dificultat per traslladar les còpies.
- Preu més elevat.

Altres dispositius

Hi ha altres dispositius que es poden utilitzar per fer còpies de seguretat. Aquests dispositius poden ser CD/DVD, discos USB, discos extraïbles o, en realitat, qualsevol dispositiu que permeti emmagatzemar dades.

Encara que es poden fer còpies de seguretat sobre aquests dispositius, no són recomanables, ja que solen fallar molt més que les solucions específiques de còpia basades en cinta o discos i els programes específics de còpia de seguretat operen de pitjor manera amb aquests.

Tot i així, en entorns molt petits o a nivell personal, poden ser una solució que s'ha de tenir en compte per fer còpies de seguretat.

Quins dispositiu hem de triar?

Dins dels dispositius que hem presentat, hauríem d'avaluar diferents paràmetres per triar entre uns dispositius o altres. Podríem respondre les preguntes següents:

1. Quin és el més econòmic? En la taula següent en podem veure una comparativa.

Tipus	Capacitat (GB)	Preu	Preu/GB €
Cinta LTO-3	800	32	0,04
Cinta LTO-4	1600	49	0,030
Cinta LTO-5	3000	90	0,03
Cinta DDS-72	72	19	0,263
Cinta DDS-120	160	41	0,256
Cinta DDS-360	320	44	0,137
Disc USB	1000	133,29	1,33
DVD	4,7	22 10 unitats	0,468
Sistema de còpies HP de disc	9000	54287	6,031
Disc dur (IDE)	1000	73	0,073

Taula amb preus orientatius de setembre del 2011

2. Necessitem emmagatzemar moltes dades? Hem d'analitzar quantes dades hem de copiar. Probablement, si no podem emmagatzemar-ho en una sola cinta, necessitarem diverses cintes. Haurem de comprar un robot de còpies

perquè la cinta vagi canviant automàticament. O, si no, n'hi haurà prou amb un sol disc.

3. Disponibilitat de les dades? Si necessitem recuperar ràpidament la informació, hauríem de tenir les nostres còpies de seguretat en sistemes de discos.
4. Còpies simultànies de diferents fonts? Si necessitem copiar simultàniament diferents fonts, necessitarem tant un programa de còpies com uns dispositius que ens permetin fer còpies concurrents. Per realitzar això, necessitarem robots de còpies amb diverses unitats de cintes o sistemes de còpies de disc.

Tenint en compte aquestes preguntes, podríem avaluar les nostres necessitats sobre el dispositiu de còpia de seguretat que s'ha de triar. Encara que sempre cal tenir en compte que els sistemes de còpia de seguretat sobre cintes són els més evolucionats i els que econòmicament surten més rendibles.

On desar la informació?

Ja hem fet les nostres còpies de seguretat. On hem de desar aquestes còpies?

Aquest tema és probablement el punt més important dins d'una planificació de còpies de seguretat. El motiu és que no serveix qualsevol lloc per emmagatzemar les nostres dades i pot significar la supervivència de la nostra empresa en cas de desastre.

Per valorar on desar les nostres còpies de seguretat, sempre ens hem de plantejar el cas més catastròfic.

Per veure'n la importància, posarem un exemple: la nostra empresa té un centre de processos de dades (CPD), on estan tots els nostres servidors importants, com els servidors de bases de dades, servidor de correus, aplicacions, fins i tot els dispositius de còpies de seguretat, per exemple un robot de cintes.

Hem planificat perfectament el nostre sistema de còpies de seguretat amb les seves còpies completes setmanals, les incrementals, les mensuals, les anuals, i això ens permet recuperar qualsevol informació. Tot això en cintes que tenen una vida útil superior a la vida de les dades que poden contenir. Aquestes cintes han d'estar etiquetades correctament: les que toquin en aquest cicle de còpia dins del robot de còpies i les restants, en una prestatgeria dins del CPD per poder

tenir-les a mà en cas de necessitar-les per a una restauració i no haver de perdre temps a anar a un altre lloc a buscar-les.

Un dia passa un desastre i es produeix un incendi dins de l'edifici, que afecta alguns departaments. Dins d'aquests departaments, afecta el CPD i, malgrat que salten tots els sistemes contra incendi del CPD, es produeix una caiguda del sostre just a sobre del *rack* dels servidors més importants. A part de la crema del departament on s'emmagatzemen els expedients en paper de tots els nostres clients i proveïdors.

Per sort, la nostra assegurança de CPD ens proveeix de la substitució de servidors en menys de 8 hores en cas de catàstrofe i tindrem a les nostres oficines els nous servidors. Simplement, haurem de restaurar la informació i la configuració en els nous servidors de les nostres còpies completes i diferencials.

Però tenim un problema: pel sostre que ha caigut ha entrat aigua dels sistemes contra incendi de la planta superior, i això ha deixat inservible les cintes de dins del robot de còpies. A més a més, també s'han mullat i s'han danyat totes les cintes de la prestatgeria. Tenim un problema molt greu, ja que no podrem recuperar la informació de la nostra empresa.

Les còpies de seguretat no poden estar totes juntes. Es podria donar la situació que tinguem certes còpies prop del CPD per tal de poder tenir les cintes a mà per recuperar les dades més ràpidament. El que no pot passar és que totes les cintes estiguin juntes.

Respecte de les còpies de seguretat, hem de tenir una planificació amb diferents jocs de còpies (setmanals, quinzenals, mensuals, anuals) i que hi hagi diferents llocs on desar aquests jocs de cintes.

És convenient tenir sales amb armaris que protegeixin contra incendis i contra inundacions, en què tinguem uns jocs de cintes. Per exemple, prop del CPD, els jocs de cintes setmanals. Per exemple, si tenim un altre edifici, cal tenir un altre armari ignífug on s'emmagatzemin les còpies quinzenals, mensuals o anuals.

També podem contractar empreses que es dediquen a desar còpies de seguretat i que tenen instal·lacions condicionades per protegir les còpies de seguretat. Aquesta és una opció que s'ha de tenir en compte per desar les còpies anuals i mensuals, que són còpies de seguretat que s'utilitzen molt menys, per posar un exemple.

En les planificacions de còpies de seguretat hem de tenir en compte totes les possibilitats i utilitzar un cicle de còpies que ens permeti desar sempre un joc de còpies (encara que sigui més antic) en una ubicació diferent del joc de còpies que

s'està utilitzant actualment. Per exemple, si tenim quatre jocs de cintes per fer els incrementals diaris, amb la seva cinta de còpia completa setmanal, estaria bé tenir només el joc actual en el robot de còpies i la resta en una altra ubicació o en diverses ubicacions. Així, si hi ha una tragèdia com l'anterior, com a màxim «només» perdríem set dies d'informació.

Cal tenir en compte, com sempre, el pressupost de què es disposa a l'hora de planificar les còpies de seguretat. No sempre es podrà tenir tot, però cal trobar el millor equilibri entre dispositius i emmagatzematge per poder recuperar els nostres sistemes contra desastres.

Rotació de suports

Hi ha diferents estratègies de rotació de suports per protegir les nostres dades. Aquestes estratègies es diferencien pel nombre de suports que necessitarem i pel temps que aquest suport es tornarà a utilitzar. El temps que tindrem aquests suports sense tornar a utilitzar serà el temps que puguem tornar enrere per recuperar un fitxer.

Tot i així, hem de tenir clars uns conceptes:

- Les còpies de seguretat no són un mètode d'arxivament, és a dir, les còpies de seguretat només són un mètode de recuperació de dades. Si tenim dades que hem d'arxivar, utilitzarem volums d'arxivament per emmagatzemar aquestes dades i no una còpia de seguretat.
- Els suports de còpia fallen i hem de tenir en compte que els haurem de substituir per altres, per a això, seria convenient consultar amb el fabricant la vida útil dels suports que utilitzem.
- Utilitzar un mètode de rotació que ens permeti una persistència molt llarga en el temps tindrà un cost més elevat en suports i una que utilitzi pocs dispositius tindrà menys persistència.

Finalment, cal comentar que la selecció d'una política de còpies de seguretat és, juntament amb el fet de seleccionar on desem les còpies de seguretat, una de les decisions més importants en una planificació de còpies de seguretat.

En els apartats següents, presentarem les estratègies de rotació més importants.

Fill

L'estratègia de rotació fill és una de les més senzilles de les estratègies. Aquesta consisteix a tenir un joc de cinc cintes, que anomenarem C1… C5, on farem una còpia completa cada dia. En la taula següent podem veure com quedaria el joc de còpies.

	Dilluns	Dimarts	Dimecres	Dijous	Divendres
Setmana 1	C1 Completa	C2 Completa	C3 Completa	C4 Completa	C5 Completa

Taula amb l'estratègia fill

És un mètode simple però que té inconvenients:

- Es reutilitzen molt les cintes, i per tant se'n redueix el temps de vida.
- Només tenim una setmana d'històric.
- Com que són còpies completes, tardarem més a fer-les.

Avantatges:

- Aquesta estratègia pot ser econòmica i utilitzable en entorns que tinguin poques dades.

A aquesta estratègia podem afegir-hi jocs de cintes per poder tenir més històric, però per això tenim altres estratègies que funcionen millor.

Pare-fill

L'estratègia de rotació Pare-fill és una estratègia que té un cicle de rotació de dues setmanes. S'hi utilitza un joc de sis cintes (C1-C6), on divendres farem una còpia completa amb les cintes C1 i C2. La resta de la setmana farem incrementals o diferencials (sempre el mateix tipus).

Podem veure un esquema en la taula següent:

	Dilluns	Dimarts	Dimecres	Dijous	Divendres
Setmana 1	C3 Incremental/ Diferencial	C4 Incremental/ Diferencial	C5 Incremental/ Diferencial	C6 Incremental/ Diferencial	C1 Completa
Setmana 2	C3 Incremental/ Diferencial	C4 Incremental/ Diferencial	C5 Incremental/ Diferencial	C6 Incremental/ Diferencial	C2 Completa

Taula amb l'estratègia pare-fill

Aquest mètode té els inconvenients següents:

- Es reutilitzen molt les cintes, amb la qual cosa reduirem el seu temps de vida.
- Només tenim dues setmanes d'històric.
- En cas que utilitzem còpies incrementals, necessitarem una còpia completa i totes les còpies incrementals d'aquesta setmana per poder recuperar tota la informació.

Avantatges:

- Bona restauració amb vista a badades, com ara l'esborrament d'arxius o fitxers molt concrets. No és extremament difícil de planificar.

Avi-pare-fill

L'estratègia de rotació avi-pare-fill o GFS (Grandfather-Father-Son) és una estratègia de rotació que utilitzen grans empreses i grans entorns. Aquesta política utilitza cintes diàries, setmanals i mensuals (en algun cas, també anuals).

Entenent que la còpia diària és el fill, *son*, les completes de la setmana són els pares, *father*. La còpia de final de mes és l'avi, *grandfather*.

La rotació es basa en un cicle de cinc dies. En una d'elles farem una còpia completa; en la resta de còpies, tindrem diferencials o incrementals. D'aquesta manera tindrem una còpia completa a la setmana. La còpia de la quarta setmana serà la que desarem com a cinta mensual i que no es tornarà a utilitzar fins a l'any següent.

D'aquesta manera, les cintes quedaran de la manera següent:

- De C1 a C3 com a cintes setmanals que farem els divendres.
- De C4 a C14 com a cintes del mes (11 cintes, la de desembre és anual).
- C15 com a cinta anual que farem el desembre. Aquesta no la reciclarem.
- De C16 a C19 com a cintes diàries, que s'hauran de canviar amb més assiduïtat que les cintes anteriors.

Seran un total de set cintes mensuals, onze mensuals i una d'anual, que sumen 19 cintes.

Podem veure un esquema de quatre setmanes en la taula següent:

	Dilluns	**Dimarts**	**Dimecres**	**Dijous**	**Divendres**
Setmana 1	C16 Incremental/ Diferencial	C17 Incremental/ Diferencial	C18 Incremental/ Diferencial	C19 Incremental/ Diferencial	C1 Completa
Setmana 2	C16 Incremental/ Diferencial	C17 Incremental/ Diferencial	C18 Incremental/ Diferencial	C19 Incremental/ Diferencial	C2 Completa
Setmana 3	C16 Incremental/ Diferencial	C17 Incremental/ Diferencial	C18 Incremental/ Diferencial	C19 Incremental/ Diferencial	C3 Completa
Setmana 4	C16 Incremental/ Diferencial	C17 Incremental/ Diferencial	C18 Incremental/ Diferencial	C19 Incremental/ Diferencial	C5 (mensual) Completa

Taula amb l'estratègia avi–pare–fill

És un mètode que presenta els inconvenients següents:

- En cas que utilitzem còpies incrementals, necessitarem una còpia completa i totes les còpies incrementals d'aquesta setmana per poder recuperar tota la informació.
- Aquesta estratègia de rotació ens obliga a utilitzar més suports que les polítiques anteriors. Per tant, implica una inversió més elevada.
- La gestió de les cintes és més complexa que les anteriors polítiques.

Avantatges:

- A llarg termini tindrem la informació d'un any; a curt, tindrem la informació d'un mes enrere.
- Bona restauració amb vista a badades, com ara esborrament d'arxius o fitxers molt concrets. No és extremament difícil de planificar.

En aquest mètode de rotació és convenient que, com a mínim, les cintes mensuals (avi) estiguin fora de la seu, encara que també seria interessant tenir fora de la seu les cintes setmanals (pares) que no siguin de la setmana en curs.

Rotació «La torre de Hanoi»

L'estratègia «la torre de Hanoi» ve del nom del puzle del joc XIX, que explota els efectes de la combinatòria. La rotació de Hanoi aprofita aquesta característica per oferir còpies de seguretat amb una protecció de dades que és de $2^{(N-1)}$ dies. N és el nombre de jocs de suports que utilitzem. Aquesta estratègia només utilitza el tipus de còpia completa, ja que els tipus incrementals i diferencials no són possibles usar-los.

En aquesta estratègia s'utilitzarà de la manera següent. Suposem un joc de sis cintes (de C1 a C6):

- C1 s'utilitzarà per al primer dia i la resta de dies en què no s'empri cap altra cinta.
- C2 s'utilitzarà cada dos dies
- C3 s'utilitzarà cada quatre dies
- C4 s'utilitzarà cada 8 dies
- C5 s'utilitzarà cada 16 dies
- C6 es farà cada 32 dies

En la taula següent podem veure com quedaria el cicle de rotació de 32 dies per a un joc de sis cintes i en què només es fan còpies de seguretat de dilluns a divendres: (Veure taula següent)

Per a cada nova cinta que introduirem en la rotació, s'utilitzarà cada doble de dies de l'anterior (32, 64, 128, 256) amb deu cintes, que és un cicle de 512 dies, equivalent a gairebé dos anys (si nomçes es realitza de dilluns a divendres).

	Dilluns	Dimarts	Dimecres	Dijous	Divendres
Setmana 1	C1 Completa	C2 Completa	C1 Completa	C3 Completa	C1 Completa
Setmana 2	C2 Completa	C1 Completa	C4 Completa	C1 Completa	C2 Completa
Setmana 3	C1 Completa	C3 Completa	C1 Completa	C2 Completa	C1 Completa
Setmana 4	C5 Completa	C1 Completa	C2 Completa	C1 Completa	C3 Completa
Setmana 5	C1 Completa	C2 Completa	C1 Completa	C4 Completa	C1 Completa
Setmana 6	C2 Completa	C1 Completa	C3 Completa	C1 Completa	C2 Completa
Setmana 4	C1 Completa	C6 Completa			

Taula amb una estratègia pare-fill

Aquesta estratègia fa un ús econòmic de les cintes i potencia la realitat que, a mesura que es remunta cap enrere en el temps, disminueix la probabilitat de recuperar una informació. Per aquest motiu, els períodes entre còpies són progressivament més llargs.

Aquest mètode presenta els inconvenients següents:

- És molt complex d'administrar.
- Com que fa còpies completes, ens cal més temps i més espai en els suports que les altres estratègies que utilitzen còpies incrementals o diferencials.

Avantatges:

- Amb poques cintes podem tenir històrics molt grans.
- Com que fa còpies completes, en qualsevol cinta tindrem totes les dades d'aquests dies i amb una sola cinta podrem recuperar els sistemes.

Seguretat en l'accés a les còpies

A part de fer les còpies de seguretat, tenir una estratègia de rotació d'acord amb les necessitats de la nostra empresa, un lloc segur contra les catàstrofes na-

turals, hauríem de tenir en compte una última cosa: protegir les nostres còpies de personal no autoritzat. En realitat, les còpies i tota la informació sensible de la nostra empresa.

Quan parlem de personal no autoritzat no ens referim a persones que no hagin de tenir accés a les còpies, sinó que ens referim al fet que en un pla de còpies de seguretat hauria d'estar perfectament indicat qui pot accedir a les còpies, qui s'ha d'encarregar de desar-les, les persones encarregades de fer tant les còpies com les restauracions. És a dir, s'han d'especificar les persones que tenen accés a les còpies.

Les còpies les han de desar les persones indicades en el pla de còpies de seguretat. Aquestes són els responsables de seguir les indicacions del pla de còpies i són les responsables de les fallades que es puguin produir. L'accés als suports de còpies ha de ser autoritzat pel responsable de seguretat i no per qualsevol persona, encara que sigui el director de l'empresa.

Encara que puguin semblar molt estrictes aquestes regles, es pot donar que en les còpies de seguretat hi hagi informació molt sensible. Podem posar un exemple:

Imaginem que treballem per a un hospital i en els nostres sistemes informàtics tenim una aplicació amb tots els historials clínics de les persones. Aquesta informació és molt sensible i hauríem de tenir-hi controlat l'accés.

En la mesura que es pugui, no s'haurien de flexibilitzar els comportaments respecte a les còpies de seguretat, ja que això ens porta al risc que succeeixin accidents o, simplement, que no es compleixin les lleis.

Les còpies de seguretat en les empreses modernes conté tota la informació necessària i imprescindible per a la continuïtat del negoci. Si es perd la informació dels servidors i no som capaços de restaurar-la, podem posar en perill la viabilitat de la nostra empresa. O en cas d'un robatori de les nostres còpies de seguretat per part de la competència, accedir als nostres secrets industrials.

Al final, l'important és l'objectiu següent: la continuïtat del negoci.

Integritat de les còpies

Dins d'una planificació de còpies hem de tenir algun mecanisme que ens asseguri que les nostres còpies estan ben fetes. Per aquest motiu, haurem de tenir una planificació de restauracions aleatòries, per poder assegurar que podem restaurar la informació.

Per fer aquestes proves aleatòries es recomana seguir les recomanacions següents:

- Revisar els *logs* del programa de còpies de seguretat per verificar que no ens dóna cap error.
- Un dia per cada cinta de què haguem de fer la recuperació d'un fitxer aleatori, per assegurar-nos que som capaços de restaurar alguna cosa d'aquesta cinta.
- Si disposem d'una base de dades, una vegada a la setmana s'ha de restaurar una d'aquestes base de dades en algun servidor auxiliar i l'arrencarem.

Aquestes proves ens donaran una mostra que la cinta o el suport on hem copiat la informació es pot restaurar.

La realització d'aquestes proves les ha de fer el departament d'informàtica i tenen inconvenients:

- Requereix diàriament temps de l'equip d'informàtica.
- Les restauracions de base de dades o de sistemes sencers requereixen una infraestructura addicional, que no sempre és possible tenir.

La revisió diària de la integritat de les nostres còpies ens assegura que estem preparats per fer correctament les restauracions. És un dels treballs més tediosos dins de la planificació de còpies de seguretat, però no es pot deixar de fer.

Restauració de les còpies

La restauració de les dades és el motiu pel qual fem una planificació de còpies de seguretat. Com hem deixat patent, la realització de còpies de seguretat té com a objectiu fer front a qualsevol pèrdua de dades i poder mantenir la continuïtat del negoci. Per la qual cosa, si hem fet correctament les còpies, ens permetrà restaurar les dades que s'han eliminat o danyat i tornar a l'estat anterior al desastre.

Tot seguit exposarem els casos més usuals, on ens veurem obligats a restaurar: restaurar un fitxer o una carpeta o restaurar un sistema des de zero.

Per fer front a la restauració de fitxers o carpetes, hauríem de conèixer la ubicació en l'estructura de directoris. Aquesta ens l'hauria de proporcionar la persona que ens ha demanat la restauració. Una vegada que en coneixem la ubicació, utilitzarem el programa de còpies de seguretat per revisar el catàleg on trobarem

les versions que tenim disponibles i les cintes que necessitem per realitzar la restauració. Aquestes situacions se solen donar per les badades dels usuaris o per algun procés que no ha tingut el funcionament esperat.

Per restaurar un sistema operatiu haurem de recuperar de l'última còpia completa i les incrementals fins a l'últim dia disponible. Segurament, de primer haurem d'instal·lar un sistema operatiu sense modificar i el client del programa de còpies de seguretat, i després haurem de restaurar el sistema complet. Aquesta situació sol passar quan tenim algun problema amb els suports d'emmagatzematge d'un servidor, com per exemple que falli un disc o un RAID al complet.

Tenint en compte això, abans de fer una restauració hauríem de tenir en compte el següent:

- Si cal tenir l'autorització del responsable de seguretat o del responsable de la informació que té aquest fitxer.
- Revisar si el destí de la restauració és diferent de l'origen i té les mateixes versions dels programes.
- Assegurar que el que volem restaurar és el que hem de restaurar.
- En cas de voler restaurar per tornar a una versió anterior, no hem de sobreescriure mai en el fitxer a l'hora de restaurar. Caldria recuperar en una altra informació perquè la persona que ho ha sol·licitat revisi que realment és el que vol.
- Una vegada recuperada la informació, cal retornar el suport al seu lloc correcte.

Conclusions

En aquest tema hem presentat un pla de còpies i el que significa per a la nostra empresa el fet de disposar o no d'aquest pla.

Resumint, en un pla de còpies haurem de decidir els punts següent:

- Quin suport utilitzarem?
- Quina estratègia de rotació i quin tipus de còpies?
- On desarem les còpies?
- Qui i com pot accedir a les còpies?

La decisió correcta de totes aquestes preguntes ens porta a disposar d'un pla de còpies que compleixi totes les nostres necessitats i sigui capaç de restaurar la informació quan calgui.

3.11. Plans de contingència

Introducció

Com podem predir, hi haurà un moment en què alguna cosa fallarà i hem de tenir previst què s'ha de fer en aquest cas. Tractarem el concepte de pla de contingència, que engloba els mètodes que s'han d'utilitzar en cas d'error.

Un pla de contingència en una empresa hauria d'estar definit per tots els processos crítics d'aquesta. És a dir, qualsevol procés crític que pugui fer que l'empresa perdi diners o trenqui hauria de tenir un pla de contingència. Ens centrarem en els plans de contingència associats als sistemes d'informació.

Posarem un exemple per veure la necessitat d'un pla de contingència:

Imaginem que treballem en el departament d'informàtica d'un banc i resulta que aquest banc té tots els sistemes informàtics en un únic CPD. Si s'esdevé un problema en aquest CPD, per exemple un terratrèmol que talla les comunicacions i destrueix el CPD completament, el banc no podria continuar el seu negoci. Això els faria perdre molts diners i, fins i tot, arruïnaria al banc.

Si tinguessin un pla de contingència adequat, per exemple tenir els sistemes crítics replicats en un altre CPD, el banc podria seguir operant des dels sistemes de contingència i seguir amb el seu negoci.

En l'actualitat, tots els bancs tenen definits plans de contingència per als seus sistemes importants, és més, molts tenen replicats els CPD en plaques tectòniques diferents, perquè el cas anterior no ocorri.

Finalment, no hem de confondre plans de contingència amb alta disponibilitat, encara que pugui ser similar, un sistema tradicional de clúster d'alta disponibilitat o de balanceig de càrrega no substituirà un pla de contingència.

Definició i propòsit d'un pla de contingència

Podríem definir un pla de contingència com un conjunt de mesures tècniques, humanes i organitzatives per garantir la continuïtat de negoci i les operacions de la companyia. És a dir, és un pla que, en cas d'algun tipus de catàstrofe, ens permetrà recuperar-nos-en.

El pla de contingència està orientat a la continuïtat de negoci, és a dir, no necessàriament el pla de contingència ha d'incloure tots els sistemes informàtics

de l'empresa. Només definirem els serveis que són imprescindibles perquè l'empresa pugui continuar funcionant. L'elecció d'aquests serveis s'ha de definir dins del pla de contingència.

Cal tenir en compte que mantenir un pla de contingència comporta unes grans despeses associades, ja que, en molts casos, requereix duplicar sistemes, comunicacions, dispositius d'emmagatzematge...

Un pla de contingència ha de ser capaç de cobrir:

- Una planificació de còpies de seguretat i recuperació capaç de reprendre els serveis informàtics de l'empresa en cas de qualsevol mena de problema.
- En cas de tenir un problema en les instal·lacions habituals, tenir capacitat de seguir donant servei des d'una altra ubicació.

En definitiva, el propòsit d'un pla de contingència és garantir la continuïtat de negoci.

Elaboració d'un pla de contingència

En els apartats següents veurem els passos per elaborar correctament un pla de contingència. Els punts que s'hi han de tractar són:

- Anàlisi de l'impacte.
- Anàlisi dels perills i l'impacte.
- Recomanacions al pla de contingència.
- Creació de les estratègies de contingència.

Anàlisi de l'impacte en l'empresa

L'anàlisi de l'impacte en l'empresa és una de les claus en la implementació de plans de contingència. L'anàlisi d'impacte proporcionarà una visió global dels processos crítics, els recursos i la relació entre els sistemes d'informació. Amb aquesta informació podrem avaluar les conseqüències de la interrupció del servei. A més a més, es podran determinar els requisits per al pla de contingència i les seves estratègies de recuperació.

Els passos que s'han de fer són:

- Determinar els processos i sistemes crítics. En aquest punt s'analitzarà l'impacte de cada procés i sistema dins de l'empresa. També identificarem per a cada sistema el temps màxim que pot estar aturat i les dades que es poden perdre.
- Identificar els recursos. Farem un inventari de tots els sistemes i les seves funcions.
- Identificació de les prioritats de la recuperació. Després d'analitzar els punts anteriors podrem establir els sistemes que són crítics i que s'han de recuperar primer, i d'aquesta manera establir un ordre de recuperació.

Determinar els processos i sistemes crítics

Dins de l'empresa hi ha sistemes i processos que fan múltiples tasques, tenen interrelació entre si. Per a tots els processos i sistemes de l'empresa, el coordinador del pla de contingència haurà d'elaborar un informe de la seva importància i criticitat. Aquest informe l'ha de fer juntament amb els responsables de l'empresa i els responsables de cada sistema que sigui crític. Aquest informe ha d'analitzar els sistemes amb criteris com disponibilitat, integritat, confidencialitat i establir un nivell d'impacte per al sistema analitzat segons aquests.

Els nivells d'impacte són els següents:

- Baix: significa que, si hi ha una pèrdua d'aquests sistemes, poden provocar degradació de la capacitat d'operacions de les empreses, però que aquesta pot continuar fent les seves funcions de manera normal i/o que es produeixen danys menors en els actius de l'empresa i/o provoca pèrdues financeres menors.
- Mitjà: pot significar que es produeix una degradació important en les operacions de l'empresa, encara que aquesta és capaç de continuar operant, l'eficiència d'aquestes operacions es redueix significativament i/o es produeixen danys greus en els béns de l'empresa i/o es produeixen pèrdues econòmiques significatives.
- Alt: pot significar que es produeix una interrupció d'un sistema que impedeix que l'empresa operi de manera normal i/o provocar danys greus en els béns de l'organització i/o es produeixen pèrdues econòmiques irreparables.

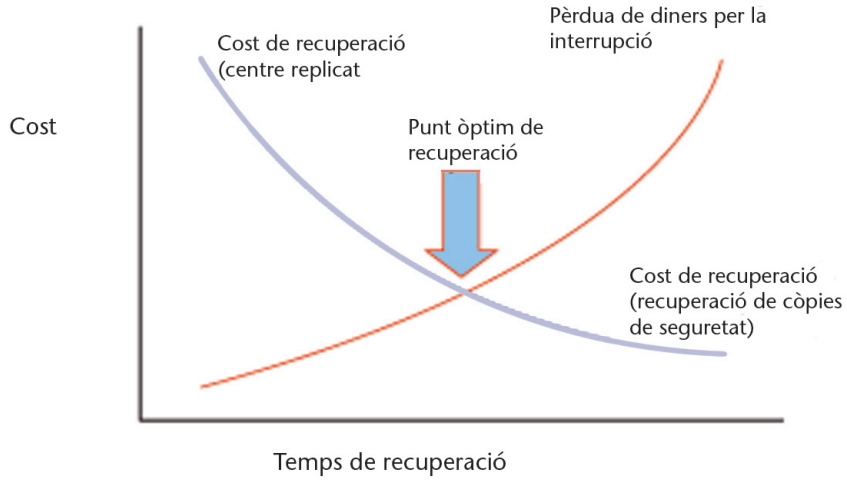
Aquests nivells d'impacte ens indicaran quins sistemes seran els més importants dins del pla de contingència.

Després d'analitzar els nivells d'impacte, el coordinador del pla de contingència haurà d'analitzar, juntament amb els responsables dels sistemes, coordinadors i gerents els temps acceptables d'inactivitat (*downtimes*) per a cada sistema. Els temps d'inactivitat s'analitzen amb els paràmetres següents:

- Temps màxim d'inactivitat tolerable (MTD, Maximum Tolerable Downtime) és el temps màxim d'inactivitat que és acceptable per part de l'empresa. Aquest temps és important, perquè ens permet triar un pla de recuperació adequat.
- Temps de recuperació objectiu (RTO, Recovery Time Objective) és el temps objectiu en què un sistema ha d'estar recuperat abans que provoqui impacte en altres processos o sistemes de l'empresa i en l'MTD. Quan no es pot complir l'RTO i l'MTD és inflexible, s'ha de començar a aplicar algun procediment definit de mitigació de l'impacte.
- Punt de recuperació objectiu (RPO, Recorey Point Objective) és el punt en el temps tolerable on es pot tornar enrere en les dades. És a dir, quant temps cap enrere ens podem permetre recuperar les dades (si l'última còpia de seguretat és de fa només dues hores, no hi ha pèrdua de dades).

Normalment, l'RTO és menor que l'MTD.

Una vegada determinat tot això, el coordinador s'ha de reunir amb els gestors de l'empresa per determinar el punt d'equilibri entre el cost de la solució de recuperació i la pèrdua econòmica produïda per la inactivitat. En la figura següent es pot veure un exemple del punt de recuperació:



Un temps excessiu en la recuperació dels sistemes ens pot portar a pèrdues econòmiques irreparables i costos de recuperació massa cars poden ser inacceptables per a l'empresa. L'ideal és combinar tots els sistemes de recuperació segons el nivell d'impacte, l'MTD, RTO i RTP.

Identificació dels recursos

El primer que cal fer dins un pla de contingència és elaborar un inventari dels recursos utilitzats en els sistemes d'informació.

En aquest inventari és convenient que s'identifiquin els recursos i quina funció fan. La taula següent pot ser un exemple:

Recurs/Component	Plataforma/SO/Versió	Funció
Servidor web	HP DL360/Linux Red Hat/5.4	Servidor web de la intranet

Taula amb un inventari de recursos

Identificació de les prioritats de la recuperació

L'assignació de prioritats en la recuperació s'ha d'assignar segons la criticitat dels sistemes. Aquesta criticitat s'estableix amb els paràmetres que hem anat assignant a cada sistema (nivell d'impacte, MTD, RTO, RPO).

Amb aquest resultat, el coordinador del pla de contingència ha de triar els diferents mètodes de recuperació i la seva prioritat per a cada sistema.

Anàlisi dels perills i l'impacte

Qualsevol empresa pot patir algun desastre o un altre tipus d'interrupció i que això passi pot tenir impacte en l'organització. Cal tenir en compte la possibilitat que s'esdevinguin desastres, sempre ens haurem de posar en el pitjor dels casos.

L'esdeveniment d'algun dels desastres ens provoca interrupcions en els nostres sistemes informàtics i haurem d'estar segurs que els nostres recursos crítics poden continuar funcionant després d'aplicar el nostre pla de contingència.

Els perills poden ser:

- Foc
- Inundacions
- Terratrèmols
- Error de subministrament elèctric
- Atacs terroristes
- Interrupcions organitzades o deliberades (atacs als sistemes)
- Fallades de l'equip
- Virus informàtics
- Error humà

En definitiva, qualsevol perill que pugui deixar els nostres serveis crítics danyats.

El pla de contingència ha de considerar totes aquestes situacions que poden impactar en les instal·lacions o els sistemes crítics del negoci.

Una vegada identificats els riscos, cal analitzar l'impacte que tindran en l'empresa. Per avaluar això, hem de tenir un bon coneixement de l'empresa, dels serveis del negoci crítics i dels recursos que utilitzen aquests serveis. Aquestes anàlisis les ha de portar a terme el departament d'informàtica i els departaments involucrats en els sistemes crítics de l'empresa.

Recomanacions al pla de contingència

Una vegada analitzats els punts crítics i els perills dels nostres sistemes d'informació s'ha d'elaborar una guia amb recomanacions sobre com protegir aquests sistemes, incloent-hi parts específiques de protecció de les dades.

En aquest punt, s'han de fer recomanacions sobre els punts següents:

- Protecció de la informació: sistemes antiintrús (tallafocs, antivirus), replicació de sistemes d'emmagatzematge de xarxa SAN i NAS (Storage Area Networkm i Network Artatched Storage), es proposarà una planificació de còpies de seguretat, que estarà relacionada amb un pla de recuperació de desastres.
- Protecció de la infraestructura del CPD, com sistemes d'alimentació ininterrompuda (SAI), sistemes contra incendi, etc.
- Protecció dels sistemes de xarxa: redundància d'infraestructura de xarxa.

Un cop identificades, aquestes recomanacions ens serviran per elaborar un pla de recuperació de desastres.

Creació de les estratègies de contingència

L'empresa ha de mitigar l'impacte que pugui provocar una catàstrofe i per a això s'han de crear estratègies de contingència. Aquestes estratègies de contingència cobriran tots els punts de còpies de seguretat, recuperació, plans de contingència, proves i manteniment.

Còpies de seguretat i pla de recuperació

Els mètodes i les estratègies de còpies de seguretat i pla de recuperació ens poden permetre restaurar els sistemes de manera ràpida i efectiva després d'una interrupció del servei. Aquestes estratègies estan enfocades a l'impacte en les interrupcions i als temps d'inactivitat dels sistemes que s'han identificat en l'anàlisi d'impacte. Aquestes estratègies s'haurien d'integrar dins de l'arquitectura de sistemes de l'empresa. És convenient avaluar tots els plans de recuperació, ja que cadascun ens servirà per a diferents sistemes i en diferents ocasions. El nivell d'impacte i els índexs de MTD, RTO i RTP ens poden indicar quina estratègia s'ha d'utilitzar.

Hem d'avaluar totes les opcions quan desenvolupem i comparem les estratègies, tenint en compte el cost, el temps màxim d'inactivitat, seguretat, prioritats de recuperació i d'integració amb l'arquitectura dels sistemes. En la taula següent veiem una relació entre nivell d'impacte, prioritat, estratègia de còpia de seguretat i pla recuperació:

Nivell d'impacte	Prioritat	Estratègia de còpia de seguretat/pla de recuperació
Baix	Baixa prioritat: impacte baix en l'empresa.	Còpia: cinta recuperació: *cold site*
Mitjà	Mitja prioritat: impacte important en les operacions de l'empresa.	Còpia: cinta, replicació de dades per xarxa. Recuperació: *warm site*
Alt	Alta prioritat: implica un impacte molt alt o irreparable en l'empresa.	Còpia: replicació de dades entre cabines de discos. Recuperació: *hot site o mirrored site.*

Taula amb els métodes de còpia y recuperació

Mètodes de còpies de seguretat

L'elecció del mètode i la política de seguretat és important dins d'una estratègia de contingència.

L'estratègia de còpia de seguretat l'elegirem tenint en compte el que hem vist en el tema anterior. Aquesta s'ha d'adaptar als requisits següents:

- Còpies de seguretat amb una periodicitat adequada a la criticitat de les dades.
- Tenir ubicats jocs de còpies en llocs alternatius on es pugui produir el desastre.
- Seguretat d'accés a aquestes cintes.

Pla de recuperació

Tenim implantades les mesures de seguretat recomanades per assegurar les nostres instal·lacions i els nostres sistemes d'informació. Tenim el nostre sistema de còpies implantat.

Però, què passa si té lloc algun desastre? Doncs, en aquest cas, hauríem de passar al nostre pla de recuperació de desastres.

I què té un pla de recuperació de desastres? Un pla de recuperació de desastres és una guia de procediments per a la recuperació dels sistemes d'informació, en cas de desastre, en el menor temps possible.

Un pla de recuperació de desastres ha de cobrir la recuperació de:

- Dades
- Maquinari
- Programes

Per a cada tipus de sistemes s'estableixen diferents tipus de procediments. No és el mateix un procediment per a un sistema crític que per a un sistema de desenvolupament. Cada estratègia de recuperació de desastres tindrà un cost, un temps de recuperació. Aquestes estratègies es poden definir com una única per a tots els sistemes o com una barreja d'aquestes, però per a empreses molt grans, amb molts sistemes i grans quantitats d'informació amb sistemes crítics, és imprescindible l'existència d'una instal·lació de suport.

Els tres tipus d'instal·lacions de suport més comuns són:

- *Hot sites*: els *hot sites* són centres de processos de dades totalment configurats i llestos per operar en poques hores després de la decisió d'entrar en contingència/suport. El maquinari, els sistemes operatius i la versió dels programes han de ser iguals que els seus homòlegs del CPD primari. També es disposa de còpies de seguretat recents. Els costos associats als *hot sites* són generalment alts, però estan justificats en aplicacions crítiques. Aquestes instal·lacions poden ser de tercers. No necessàriament ha de ser una instal·lació pròpia.

  Els *hot sites* estan pensats i dissenyats per operar en cas d'emergència durant un temps limitat i no per a un ús habitual a llarg termini. Com a conseqüència, el *hot site* s'ha de considerar com una forma de continuar amb el negoci per un termini de temps determinat en l'espera que el CPD primari es restableixi.

  Aquests centres de suport, normalment els utilitzen empreses financeres, proveïdors de comerç electrònic; al cap i a la fi, empreses o institucions per a les quals un minut de RTO implica una pèrdua de diners important. També és important per a institucions que disposen d'informació molt important i que han d'estar sempre accessibles; per exemple, imagi-

nem el sistema informàtic que contingui els historials clínics de tota una regió, si no es disposa d'accés a ells, no es podrà atendre correctament els malalts.

- *Cold sites*: són CPD que només tenen la infraestructura bàsica, és a dir, cablejat elèctric, aire condicionat, etc., però que no disposen de cap component de maquinari instal·lat.

  Aquest està preparat per rebre el maquinari necessari per començar a recuperar els sistemes. L'activació d'aquests tipus de centres pot portar molt temps, sempre segons la mida de la infraestructura que tingui l'empresa.

  Aquest tipus de centres de suport està pensat si disposem de diners molt limitats, la nostra infraestructura és petita (es pot recuperar en dies) i si la nostra empresa es pot permetre tenir RTO de dies.
- *Warm sites*: són un CPD intermedi entre un *hot site* i un *cold site*. Aquest tipus de centres de dades estan totalment configurats per poder operar al cap de poques hores des que té lloc el desastre. La diferència amb els *hot sites* és que els sistemes estan en una escala més petita que en el centre de processos de dades principal i, de vegades, no tenen tots els sistemes que hi ha al centre principal. Només els més crítics.

  Aquest tipus de centre de dades ens proporcionarà tenir els sistemes disponibles en poc temps, encara que amb un rendiment menor. A causa d'això, el temps d'operació és totalment temporal i és adequat recuperar el centre principal com més aviat millor per poder operar amb normalitat.

  Els centres de suport *warm sites* són més econòmics que els *hot sites* i ens proporcionen una funcionalitat similar, encara que un rendiment menor.

  Dins d'aquests tipus d'instal·lacions hi ha variacions i barreges entre elles. D'aquestes variants hi ha dos que són remarcables:

  –*Mobile sites*: són centres de processos de dades autocontinguts, és a dir, en un continent que conté telecomunicacions i tot el maquinari necessari per al seu funcionament.

  –*Mirrored sites*: són centres de processos de dades similars als *hot sites*, amb la característica que les dades estan sincronitzades entre els dos centres en temps real. En aquest tipus de centres poden no tenir tots els sistemes replicats. És un tipus de centres de suport que s'utilitza amb sistemes que són extremament crítics.

En la taula següent podem veure una comparativa de les instal·lacions de suport:

Instal·lació	Cost	Maquinari	Xarxa	Temps de resposta
Cold Sites	Baix	Cap	Cap	Llarg
Warm Sites	Mitjà	Parcial/Tot	Parcial/Tot	Mitjà/Baix
Hot Sites	Alt	Tot	Tot	Baix

Taula comparativa de centres de suport

L'elecció d'un centre de suport s'haurà de basar en el cost contra el benefici que ens proporcioni.

Identificació dels equips responsables i les funcions

Una vegada triats i implementats els nostres plans de suport, caldria assignar un coordinador del pla de contingència. També s'han d'assignar els equips responsables de cada funció. Dins de cada equip s'han de delimitar funcions a cada persona i el procediment que s'ha de portar a terme.

Una divisió d'equips de sistemes d'informació per als plans de contingència i recuperació són:

- Equip de gestió (que ha d'incloure el coordinador del pla de contingència).
- Equip d'avaluació de la interrupció.
- Equip d'administració de sistemes operatius.
- Equip d'administració d'aplicacions.
- Equip d'administració de xarxa.
- Equip de recuperació de base de dades.
- Equip de recuperació d'operacions de xarxa.
- Equip de proves.
- Equip de seguretat.

Les persones que componguin aquests equips han de tenir habilitats i coneixements de cada equip. Idealment, les persones haurien de ser les que normalment tinguin unes funcions similars en condicions normals. Els components dels equips han de comprendre el propòsit del pla de contingència i conèixer els

procediments que hi han d'aplicar. Aquests equips han de tenir la mida suficient per si de cas algun dels integrants no està disponible. Dins de cada equip és convenient que conegui els procediments dels altres equips perquè hi hagi una bona interacció entre els equips. El coordinador s'ha d'encarregar de tenir en compte el personal que pot tenir problemes en cas d'interrupció i posar remei buscant-li substituts. Per aquest motiu, hi ha d'haver equips de reserva que tinguin coneixement del pla de contingència.

Dins de cada equip hi ha un cap d'equip que s'encarrega d'organitzar les operacions i els procediments de l'equip. També hi ha d'haver un cap d'equip alternatiu, per si de cas la persona designada no està disponible.

Si el sistema és prou gran, cal l'equip de gestió que s'encarregui d'activar el pla de contingència, coordinar les tasques entre els equips i supervisar el pla de contingència. El cap d'aquest equip sol ser el responsable d'informàtica de l'empresa, que és l'últim responsable de les decisions.

Proves del pla de contingència

Dins d'un pla de contingència i del cicle de vida d'aquest, una de les coses més importants que s'ha de fer són les proves que certifiquin que tot funciona com nosaltres esperem que funcioni.

Aquestes proves han de ser el més realistes possibles, és a dir, una vegada cada cert temps (com a mínim un cop l'any) s'ha de fer una prova com més real millor. Hem de simular una catàstrofe executant tot el procediment des del principi, mobilitzant els responsables i els equips, aixecant el centre de suport i operant durant el centre de suport com a mínim durant un dia.

Després de fer les proves s'ha d'analitzar com s'ha executat el procediment, detectar els possibles problemes i/o mancances que ha tingut l'empresa per operar d'una manera més o menys normal o com s'esperava.

Una vegada feta l'anàlisi del funcionament de pla de contingència, s'ha de modificar i fer una altra prova.

Una pràctica comuna en empreses que tenen centres de dades replicats és anar commutant de centre primari a secundari, i utilitzar-lo durant un temps, per garantir que el centre de suport funciona correctament.

Elaboració del procediment de contingència i cicle de vida

Una vegada realitzades les tasques dels punts anteriors, ja coneixem l'entorn, els processos i els sistemes crítics de la nostra empresa, coneixem l'impacte que es pot produir en la nostra empresa en cas de catàstrofe. Hem fet unes recomanacions per evitar, en la mesura que es pugui, els efectes d'una catàstrofe. Coneixem la forma d'actuar en cas d'un desastre, els responsables i les seves funcions. També sabrem com n'és de sensible l'empresa davant el que s'ha proposat i si hi ha destinat els recursos econòmics necessaris.

Amb tota la informació recollida en els punts anteriors estem en el punt d'elaborar el manual de contingència, que és un informe de tota la informació que hem recopilat. El document ha de ser simple, comprensible, sense ambigüitats i s'hi han d'especificar els punts següents:

- Què hi ha abans del desastre.
- En quines situacions declarem que s'ha produït un desastre.
- Quins sistemes d'informació s'han de recuperar.
- Identificació dels responsables.
- Un pla d'acció.
- Procediment de recuperació (pla de recuperació de desastres).

Una vegada elaborat, el pla de contingència s'ha de sotmetre a una avaluació contínua. Cada vegada que s'afegeix alguna cosa als sistemes d'informació de l'empresa, s'ha d'avaluar de quina manera ha d'estar reflectit dins del pla de contingència i del pla de recuperació de desastres. El pla de contingència no és estàtic, ha d'evolucionar tenint en compte les proves que es realitzin i les conclusions a què s'arribi d'aquestes proves.

També hem de tenir en compte les necessitats canviants de la nostra empresa, el que un dia és crític, demà no ho és i viceversa. Un sistema que avui no és crític, demà és la clau que la nostra empresa aguanti una catàstrofe. Per aquest motiu hem d'estar avaluant constantment el nostre pla de contingència.

Els plans de contingència han de seguir un cicle de vida PDCA, acrònim de *Plan*, *Do*, *Check*, *Act* (planificar, fer, verificar, actuar). Bàsicament, és un iteració contínua on planifiquem el nostre pla de contingència, l'implementem, verifiquem que funcioni correctament i, si no funciona, doncs refem el pla.

Conclusions

En aquest tema, juntament amb les planificacions de còpies de seguretat, hem presentat una manera diferent de veure la seguretat, des d'un punt de vista de continuïtat de negoci.

La importància dels plans de contingència i de les còpies de seguretat és poder continuar amb el nostre negoci. En cas de catàstrofe i/o pèrdua de dades, poder restaurar els nostres sistemes d'informació per poder continuar treballant amb la pèrdua mínima de dades.

En definitiva, hem deixat patent amb aquests temes que és tan important la seguretat dels nostres sistemes com tenir un pla de còpies i de contingència.