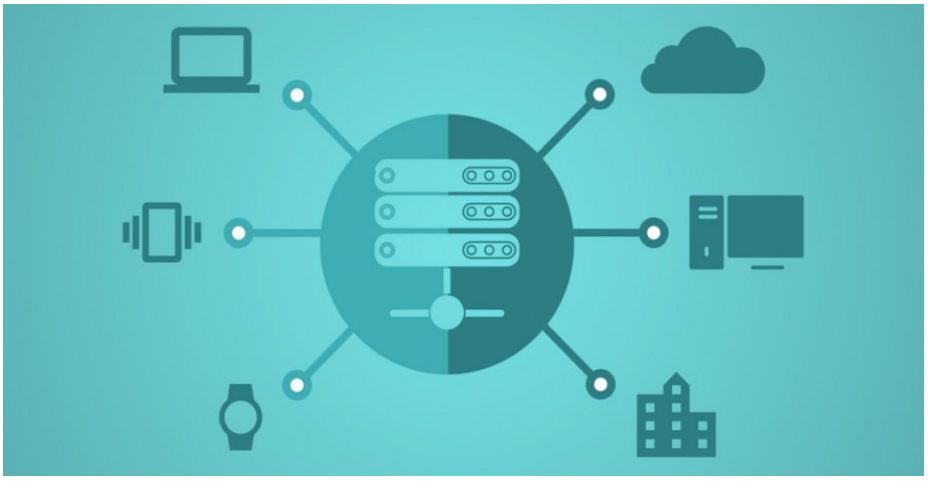
Mòdul professional:

Desplegament d'Aplicacions Web

Codi: 0614

Durada: 80 hores



Continguts:

Tema 1. Implantació d'arquitectures web

Tema 2. Administració de servidors web

Tema 3. Administració de servidors d'aplicacions

Tema 4. Instal·lació i administració de servidors d'FTP

Tema 5. Serveis de xarxa implicats en el desplegament d'una aplicació web

Tema 6. Documentació i sistemes de control de version

Xavi Blanes 2024

ORDRE: Desplegament d'Aplicacions Web | Codi: 0614

T-1: Implantació d'arquitectures web:
– Aspectes generals d'arquitectures web
– Escalabilitat, portabilitat, componentització, patrons de disseny, entre altres.
– Arquitectures web. Models.
– Plataformes web lliures i propietàries.
– Servidors web i d'aplicacions. Instal·lació i configuració bàsica.
– Tipus, model de programari lliure i programari propietari.
– Instal·lació de servidors web de programari lliure i programari propietari.
– Estructura i recursos que componen una aplicació web. Descriptor de desplegament.

T-2: Administració de servidors web:
– Característiques generals d'un servidor web.
– Protocol HTTP.
– Tipus MIME.
– Configuració avançada del servidor web.
– Fitxers de configuració i/o ferramentes de configuració.
– Mòduls: instal·lació, configuració i ús.
– Amfitrions virtuals. Creació, configuració i utilització.
– Autenticació i control d'accés.
– El protocol HTTPS.
– Configuració segura de servidors http.
– Certificats. Servidors de certificats.
– Navegadors web. Paràmetres d'aparença i ús.
– Desplegament d'aplicacions sobre servidors web.
– Proves de funcionament de l'aplicació web.

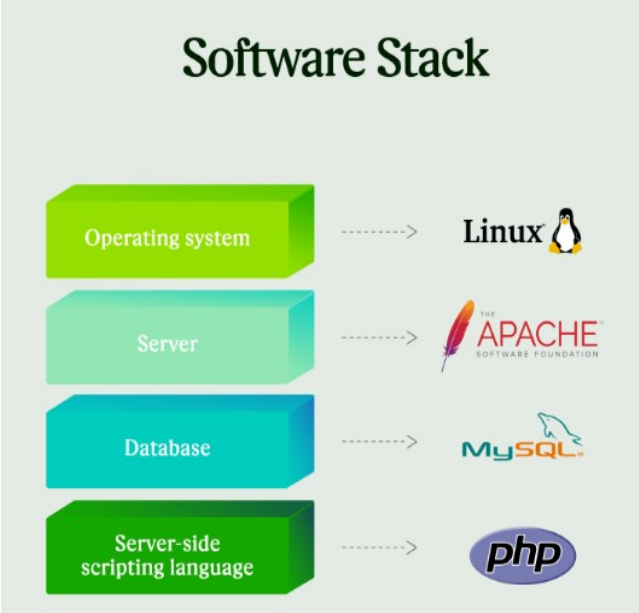
T-3: Administració de servidors d'aplicacions:
– Configuració i ús dels components web.
– Tecnologia bàsica de servicis web.
– Protocols.
– Llenguatges de descripció de servicis web.
– Arquitectura i configuració bàsica del servidor d'aplicacions.
– Administrar aplicacions web.
– Autenticació d'usuaris. Dominis de seguretat per a l'autenticació.
– Administració de sessions. Sessions persistents.
– Arxius de registre d'accés i filtre de sol·licituds.
– Configurar el servidor d'aplicacions per a cooperar amb servidors web.
– Desplegament d'aplicacions en el servidor d'aplicacions.
– Seguretat en el servidor d'aplicacions. Configurar el servidor amb suport SSL/T.
– Allotjament compartit del servidor d'aplicacions.

T-4: Instal·lació i administració de servidors de transferència de fitxers:
– Funcionalitat del servici de transferència de fitxers.
– Servidors i clients.
– Configuració del servici de transferència de fitxers. Permisos i quotes.
– Tipus d'usuaris i accessos al servici.
– Modes de connexió del client.
– Tipus de transferència de fitxers.
– Protocol segur de transferència de fitxers.
– Utilització de ferramentes gràfiques.
– Servici de transferència d'arxius des del servidor web.
– Utilització del servici de transferència de fitxers en mode text.
– Ordes.
– Utilització del servici de transferència d'arxius des del navegador.
– Utilització del servici de transferència d'arxius en el procés de desplegament de l'aplicació web.

T-5: Servicis de xarxa implicats en el desplegament d'una aplicació web:
– Sistemes de noms plans.
– Sistemes de noms jeràrquics.
– Resolutors de noms. Procés de resolució d'un nom de domins
– Dominis de primer nivell i successius.
– Instal·lació de servidors de resolució de noms.
– Paràmetres de configuració i registres del servidor de noms afectats en el desplegament.
– Servici de directoris: característiques i funcionalitat.
– Instal·lació de servidors de servici de directori.
– Fitxers bàsics de configuració. Interpretació i ús.
– Autenticació d'usuaris en el servici de directoris.
– Adaptació de la configuració del servidor de directoris per al desplegament de l'aplicació. Usuaris centralitzats.
– Documentació associada als processos d'adaptació dels servicis.

T-6: Documentació i sistemes de control de versions:
– Formats estàndard per a la documentació.
– Llenguatges de marques per a la documentació.
– Descripció de tipus de documents.
– Ferramentes externes per a la generació de documentació. Instal·lació, configuració i ús.
– Creació i utilització de plantilles.
– Instal·lació, configuració i ús de sistemes de control de versions.
– Operacions avançades i seguretat dels sistemes de control de versions.
– Història d'un repositori.
– Documentació sobre la instal·lació, la configuració i l'ús del sistema de control de versions

Tema 1: Implantació d'arquitectures web



Índex:

1. Que entenem per arquitectura web

Una arquitectura web, o arquitectura d'aplicació web, es refereix a la estructura i el disseny d'una aplicació o sistema que s'executa a través de la web. Aquesta arquitectura defineix com les diferents parts d'una aplicació web interactuen entre elles i com es gestiona la lògica i el flux de dades. Sol ser un conjunt de components i protocols que permeten als usuaris accedir i interactuar amb l'aplicació mitjançant un navegador web.

Les arquitectures web solen incloure els següents components principals:

1. Servidor web: És el component que allotja l'aplicació i gestiona les sol·licituds dels usuaris. Pot ser un servidor físic o virtual en un centre de dades o a la núvol.
2. Navegador web: És l'aplicació que els usuaris utilitzen per accedir a l'aplicació web mitjançant una interfície d'usuari gràfica.
3. Base de dades: Moltes aplicacions web emmagatzemen dades en una base de dades que es pot accedir i actualitzar a través de la web.
4. Llenguatges de programació: S'utilitzen llenguatges com HTML, CSS i JavaScript per crear la interfície d'usuari i gestionar la lògica de l'aplicació al costat del servidor.
5. APIs (Interfícies de programació d'aplicacions): Es poden utilitzar APIs per permetre que altres aplicacions o sistemes s'interconnecten amb l'aplicació web.

L'arquitectura web pot variar en complexitat i disseny segons les necessitats de l'aplicació. Existeixen diferents models d'arquitectura web, com ara l'arquitectura monolítica, l'arquitectura orientada a serveis (SOA), l'arquitectura de microserveis i altres, que determinen com les diferents parts de l'aplicació estan organitzades i comunicades entre elles.

En resum, una arquitectura web és el marc en el qual s'estructura i opera una aplicació web, permetent als usuaris interactuar amb aquesta a través d'un navegador web o una altra interfície de client.

2. Implantació d'arquitectures web

L'arquitectura d'aplicacions en entorns web difereix molt de la d'aplicacions d'escriptori, en la qual un programa s'executa en alguna d'aquestes modalitats: interpretat, en una màquina virtual o directament sobre la màquina en la qual treballa l'usuari.

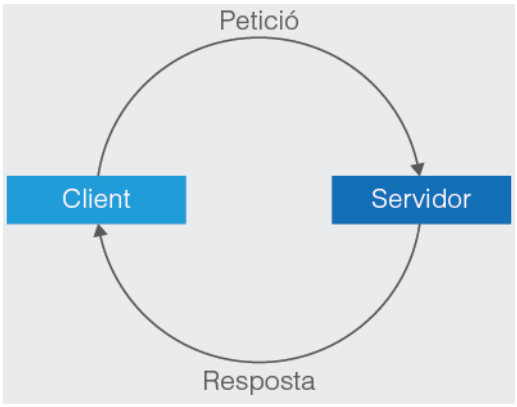
El model d'arquitectura bàsic que hi ha en tota aplicació web és el model anomenat client-servidor, en el qual entren en joc diverses màquines o plataformes, cadascuna de les quals desenvolupa un rol diferenciat en l'execució de l'aplicació. Segons les necessitats i la complexitat de l'aplicació, aquest model bàsic d'arquitectura es pot complicar més o menys per tal d'aconseguir una millor distribució de tasques, millor rendiment, fiabilitat, augment de la capacitat de procés, etc.

3. Arquitectures web. Models

Una aplicació distribuïda està formada per una col·lecció d'ordinadors autònoms enllaçats per una xarxa d'ordinadors i suportats per un programari que fa que el conjunt actue com un servei integrat.

El model client-servidor

El model client-servidor és un model d'arquitectura d'aplicacions en el qual es defineixen o s'assignen principalment dos rols als ordinadors, que són, com el nom del model indica, els rols de client i de servidor:



En el model client-servidor hi ha dos tipus de components:

- Clients: fan peticions de servei. Normalment els clients inicien la comunicació amb el servidor.
- Servidors: proveeixen serveis. Normalment els servidors esperen rebre peticions. Una vegada han rebut una petició, la resolen i retornen el resultat al client.

Els servidors poden ser amb estat o sense estat. Un servidor sense estat no manté cap informació entre peticions, mentre que un servidor amb estat pot recordar informació entre peticions. Per exemple, un servidor sense estat podria ser aquell que conté pàgines web estàtiques. En canvi, un servidor que tinga una pàgina web amb contingut dinàmic seria un exemple de servidor amb estat.

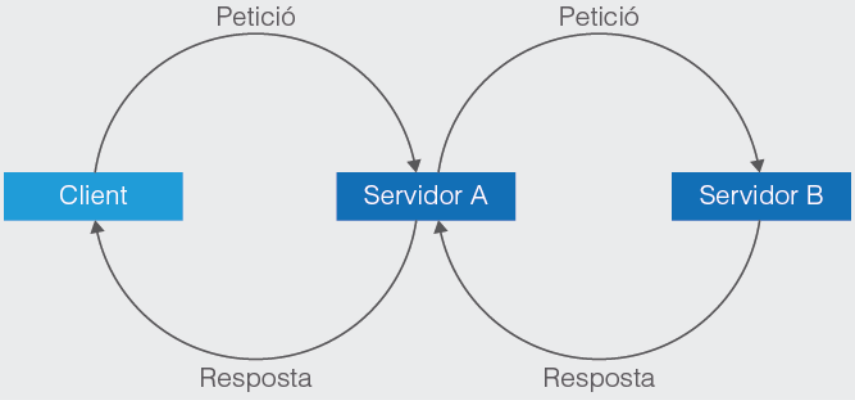
El model client-servidor bàsic de la figura anterior és vàlid per a aplicacions web xicotetes, senzilles i que no tinguen una gran càrrega de treball, és a dir, un nombre baix de clients connectats simultàniament.

Un servidor també pot ser client d'altres servidors. Per exemple, els servidors web i els altres serveis disponibles a internet són clients del servei de resolució de noms (DNS).

En entorns reals és habitual que no es donen aquestes tres característiques i, per tant, s'haja d'implementar una arquitectura més complexa basada en el model client-servidor però que pot presentar diferències o ampliacions al model per tal de garantir un bon rendiment de les aplicacions web, la seva fiabilitat i/o la capacitat d'atendre un nombre elevat de peticions dels clients de forma simultània en aplicacions web de mida mitjana o gran i d'un nivell de complexitat mitjà/alt.

El model client-servidor amb servidors encadenats

Quan en una aplicació el servidor ha de realitzar tasques molt complexes o costoses de processar poden distribuir-se subtasques en diversos servidors, de tal manera que un servidor pot actuar com a client d'un altre servidor per tal de delegar determinades responsabilitats. En la figura tenim l'structura client-servidor encadenada.



Per exemple, quan un client d'una entitat bancària accedeix als serveis en línia del seu banc amb un navegador web (client), el client inicia una sol·licitud al servidor web del banc. Les credencials d'inici de sessió del client estan emmagatzemades en una base de dades i el servidor web accedeix al servidor de base de dades com a client. Un servidor d'aplicacions interpreta les dades retornades aplicant la lògica de negoci del banc i proporciona l'eixida al servidor web. Finalment, el servidor web retorna el resultat al navegador web del client per a la seva visualització.

Aplicacions basades en el web

Un cas particular d'aplicacions client-servidor són les aplicacions que s'executen aprofitant l'arquitectura del web. Aquestes aplicacions es basen en el fet de tindrer tota la capacitat de processament en un servidor web (o conjunt de servidors) al qual s'accedeix des d'un navegador web.

Quan un usuari clica sobre un enllaç d'una pàgina web del seu navegador, aquest genera una petició al servidor que conté la informació. Una vegada el servidor rep la petició, retorna el contingut. La comunicació entre client i servidor es fa mitjançant el protocol HTTP.

El model entre iguals

Hi ha un tipus d'arquitectura en què tots els ordinadors es comporten al mateix temps com a clients i com a servidors. Aquests tipus de xarxes s'anomenen **d'igual a igual** (*peer-to-peer*).

Un sistema d'igual a igual es caracteritza per ser un sistema distribuït en què tots els nodes tenen les mateixes capacitats i responsabilitats, és a dir, tots són clients i servidors al mateix temps i, per tant, tota la comunicació és simètrica.

4. Servidors web i d'aplicacions. Instal·lació i configuració bàsica

Durant les fases de desenvolupament, de posada en producció i de manteniment d'una aplicació web podem trobar-nos amb diversos tipus de servidors que duen a terme tasques concretes dins el funcionament global.

Servidors web

Un servidor web és un servidor que permet l'accés a recursos mitjançant el protocol HTTP (*HyperText Transfer Protocol*) d'internet.

La definició original i estricta del concepte de servidor HTTP fa referència a aquells servidors capaços de donar accés i de permetre la gestió d'un conjunt de recursos estàtics com a resposta a peticions rebudes pels clients. És a dir, que permeten consultar, carregar i eliminar recursos del servidor. Aquests recursos solen ser documents d'HTML o variants d'aquest format i continguts adjunts o relacionats amb aquests documents, com poden ser imatges, vídeos, etc.

Aquests recursos solen estar guardats en forma d'arxius a dispositius d'emmagatzematge propis del servidor.

El concepte original de servidor web no contempla la possibilitat de generar de forma dinàmica els continguts a partir de l'execució de codi com a resposta de les peticions. Però, en l'actualitat, la majoria de servidors web admeten la instal·lació de mòduls que permeten que es generen continguts dinàmics a partir de l'execució de programes escrits en diversos llenguatges de programació (PHP, Javascript, Python, Perl, etc.), tot i que aquesta característica és més pròpia dels servidors d'aplicacions.

Alguns exemples de servidors web són Apache HTTP Server, per a sistemes operatius Linux, i Microsoft Internet Information Server, per a Windows.

Servidors d'aplicacions

Un servidor d'aplicacions en general és un servidor que ofereix als clients un servei d'execució d'aplicacions. Si ens centrem en les aplicacions web, un servidor d'aplicacions és un programari que controla l'execució de programes. Els clients, des d'un navegador (usant el protocol HTTP), accedeixen a una interfície web des d'on executaran l'aplicació. Normalment, els servidors d'aplicacions s'utilitzen en aplicacions web amb un grau de complexitat elevat.

Un servidor d'aplicacions web es pot entendre com un servidor orientat a l'execució de programes que pot rebre les peticions de servei i retornar els resultats utilitzant els mateixos protocols (HTTP) i formats de dades que els servidors web (HTML). Si el mateix servidor no té la capacitat d'interactuar amb aquests protocols pot treballar conjuntament amb el suport d'un servidor web que faça d'intermediari entre el servidor d'aplicacions i el client. Els servidors d'aplicacions, a més, acostumen a proporcionar un ampli conjunt de serveis complementaris orientats a la persistència de dades, la seguretat, el control de transaccions i concurrència, entre d'altres.

Alguns exemples de servidors d'aplicacions són GlassFish (servidor Java EE, Oracle) o Microsoft Internet Information Server (servidor .NET).

Servidors de bases de dades

Un servidor de bases de dades s'utilitza per emmagatzemar, recuperar i administrar les dades d'una base de dades. El servidor gestiona les actualitzacions de dades, permet l'accés simultani de molts servidors o usuaris web i garanteix la seguretat i la integritat de les dades.

Entre les seues funcions bàsiques, el programari de servidors de bases de dades ofereix eines per facilitar i accelerar l'administració de bases de dades. Algunes funcions són l'exportació de dades, la configuració de l'accés dels usuaris i el suport de dades.

Alguns exemples de servidors de bases de dades són Oracle Database, MySQL, Microsoft SQL Server, PostreSQL, MongoDB o Firebase.

Servidors de fitxers

Un servidor de fitxers és un servidor que permet gestionar a través de xarxa la càrrega, descàrrega, actualització i eliminació de fitxers emmagatzemats en els seus dispositius des d'ordinadors client.

En l'àmbit de les aplicacions web, els servidors de fitxers s'utilitzen principalment per desplegar les aplicacions sobre el servidor on s'executaran. El desplegament d'una aplicació web sobre els servidors de producció comporta habitualment la càrrega de grans quantitats de fitxers sobre aquests servidors. Com que el desenvolupament i manteniment d'aquestes aplicacions es fa en les màquines dels programadors, cal algun sistema de transferència d'arxius cada vegada que es vol actualitzar la versió de producció d'una aplicació.

Un dels protocols més usats per a la transferència de fitxers en el desplegament d'aplicacions web és el protocol FTP (*file transfer protocol*), amb les seves variants FTPS i SFTP per adaptar-se a les necessitats actuals de seguretat.

Alguns exemples de servidors de transferència de fitxers són ProFTPD o vsftpd, per a sistemes operatius Linux, i Microsoft Internet Information Server, per a Windows.

Servidors de directori

Un servidor de directori és un servidor que permet gestionar informació administrativa respecte a l'entorn d'una aplicació web, com poden ser, per exemple, els usuaris autoritzats amb els seus rols o permisos, etc.

La utilitat principal dels servidors de directori és facilitar la gestió d'informació relativa a l'explotació d'aplicacions web. L'avantatge de gestionar aquesta informació mitjançant aquest tipus de servidors és la centralització de dades i la facilitat d'accés mitjançant protocols estàndard com LDAP.

Alguns exemples de servidors de directori són OpenLDAP, per a Linux, i Active Directory, per a Windows.

Estructura i recursos d'una aplicació web

Les aplicacions web, a més de presentar un arquitectura client-servidor (fet que no és necessari en el cas de les aplicacions d'escriptori), solen estar estructurades amb una gran quantitat d'arxius i recursos de tipus diferents.

És per això que cal establir unes directrius per tal d'organitzar la ubicació d'aquests components i la seva interrelació durant la fase de desenvolupament, així com també en el moment de posar l'aplicació en producció. En cas contrari, el desenvolupament i manteniment d'una aplicació de mida mitjana o gran es convertirà en una tasca impossible de gestionar.

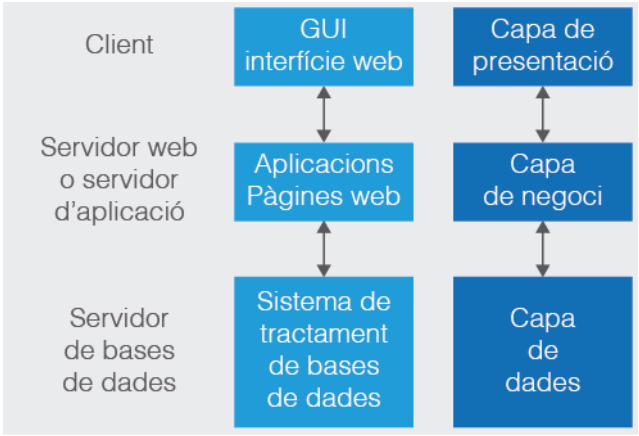
Oblidant-nos de l'organització o estructura que imposa el fet d'escollir unes determinades eines de desenvolupament o un determinat servidor web o d'aplicacions, aquestes aplicacions es poden estructurar segons diversos models d'organització dels seus components i recursos. Alguns dels models d'estructuració d'aplicacions web que podem trobar més habitualment són els que es descriuen a continuació.

Arquitectura multinivell

L'arquitectura multinivell (*multitier architecture*) és un tipus concret de l'arquitectura client-servidor en la qual els components i recursos d'una aplicació se separen segons la seua funció. Una de les divisions més utilitzades és la que separa el nivell de presentació, el nivell de lògica d'aplicació i el nivell de gestió de dades.

En aquest cas, l'estructura concreta seria de tres nivells (*3-tier architecture*). El model es defineix com a *N-tier architecture* (multinivell), ja que proposa una divisió flexible de les aplicacions en els nivells que calga per tal de fer més eficient el seu desenvolupament, manteniment i explotació.

En aquest model, la divisió per nivells es fa de forma lineal: el nivell 1 interactua de forma directa i única amb el nivell 2, el nivell 2 interactua amb el 3, i així successivament . Figura Arquitectura multinivell :



Cal diferenciar entre el concepte multinivell (*multitier N-tier*) i multicapa (*multilayer N-layer*). En aquest cas, es considera que en el model multinivell cada nivell, a més d'implementar una funció concreta, és executat per un maquinari diferent de la resta de

nivells. En el model multicapa, cada capa desenvolupa una funció concreta que pot ser executada per un mateix ordinador que s'encarrega, també, de l'execució d'altres capes.

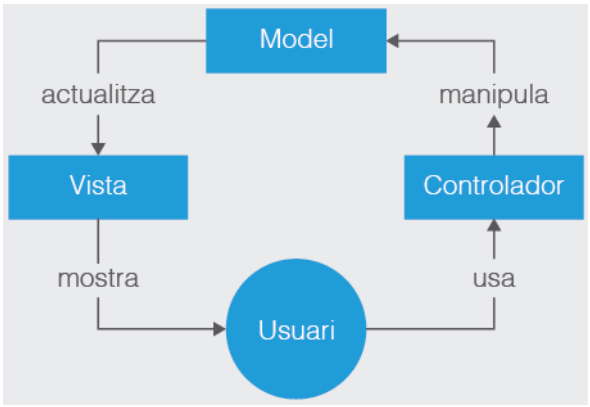
Arquitectura model-vista-controlador

L'arquitectura model-vista-controlador (*model view controller*) és una arquitectura que separa la representació de la informació i la lògica d'una aplicació de la interacció de l'usuari.

Els tres elements que defineix aquesta arquitectura són:

- Model: conté les dades de l'aplicació, les regles de negoci o la lògica de l'aplicació i les seves funcions.
- Vista: és la representació visible de l'aplicació, la sortida de les dades cap a l'usuari, és a dir, la interfície.
- Controlador: controla la interacció de l'usuari (entrada de dades) i converteix aquesta interacció en ordres o comandes per al model o la vista.

La interrelació entre els elements d'aquesta arquitectura no es fa seguint un model lineal com el model multinivell, sinó que es tracta d'un model circular. Figura Arquitectura mvc:



Paral·lelament a l'estructura de l'aplicació, cal tenir en compte que cada nivell, capa o mòdul pot estar format per un gran nombre de components i recursos de diversos tipus: fitxers HTML, CSS, imatges, etc.

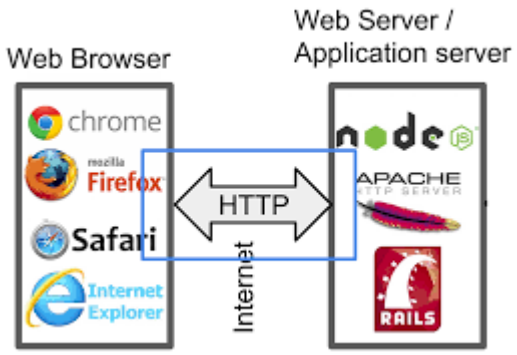
Per això és convenient establir un sistema d'organització coherent i eficient per tal d'estructurar tots aquests components que s'acaben generant durant el desenvolupament d'una aplicació web. La majoria de plataformes de desenvolupament avançades imposen mecanismes per tal d'organitzar i descriure de manera sistemàtica la localització, les característiques i la configuració dels components i recursos de les aplicacions.

Entre aquests mecanismes en destaquen dos:

- Estructura de directoris: les plataformes avançades de desenvolupament d'aplicacions web acostumen a definir una estructura de directoris mínima que tota aplicació ha de tenir a partir de la qual es despleguen els diversos tipus de components. Els desenvolupadors han de seguir les directrius de cada plataforma.

- Descriptor de desplegament: hi ha un fitxer de configuració on es pot especificar el nom, la ubicació i els paràmetres de configuració dels diversos components que formen una aplicació per tal de tenir aquesta informació centralitzada, accessible i actualitzable sense necessitat de fer modificacions en el codi font de l'aplicació. Aquest descriptor descriu com s'ha de desplegar l'aplicació en el servidor.

Tema 2: Administració de servidors web



Índex:

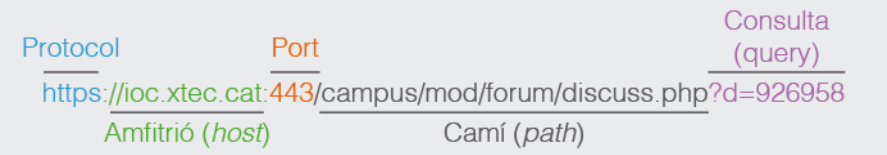
1. Introducció

Abans de començar amb la instal·lació, configuració i administració de servidors web és bo conèixer les qüestions fonamentals del protocol HTTP. El protocol de transferència d'hipertext o HTTP (*HyperText Transfer Protocol*) estableix el protocol per a l'intercanvi de documents d'hipertext i multimèdia al web. L'HTTP disposa d'una variant xifrada mitjançant SSL anomenada HTTPS. Tenint en compte que HTTP és el protocol utilitzat en les comunicacions web, conèixer els aspectes bàsics d'aquest protocol ajuda a entendre algunes de les qüestions que tenen a veure amb la configuració i administració de servidors web.

L'URL

Els localitzadors uniformes de recursos (URL, *Uniform Resource Locator*) són el mecanisme que permet, com indica el seu nom, localitzar recursos a internet mitjançant els diversos protocols disponibles. Existeix també el concepte d'identificador uniforme de recursos (URI, *Uniform Resource Identifier*) que, a la pràctica, es considera equivalent al concepte d'URL, ja que un URI equival al conjunt d'un URL més un nom uniforme de recursos (URN, *Uniform Resource Name*). Com que el concepte URN a la pràctica no s'usa, els termes URL i URI són equivalents i intercanviables en la major part dels casos.

Un URL està format per diversos elements cadascun dels quals ens descriu un aspecte diferent sobre la localització d'un recurs. Els elements en el cas dels URL utilitzats en els protocols HTTP/HTTPS són els següents:



- Esquema (*scheme*): indica el protocol que s'utilitzarà per accedir al recurs especificat per la resta de l'URL. Segons el protocol indicat (HTTP i HTTPS), la resta de l'URL pot tenir diferències en la seua estructura.
- Informació d'usuari (*user info*): aquest element no forma part de la definició estàndard dels URL, però molts servidors web contemplen l'opció d'ajuntar informació d'usuari (nom, i opcionalment *password*) per tal de facilitar processos bàsics d'identificació i validació d'usuaris.
- Allotjador (*host*): identifica el servidor web. Pot ser un nom de domini o una adreça IP.
- Port: es tracta d'un element opcional que serveix per indicar quin port TCP/IP s'utilitzarà per establir la connexió per accedir al recurs. Amb protocol HTTP, si no s'indica, agafa per defecte el port 80 i amb HTTPS s'utilitza per defecte el port 443.
- Camí (*path*): indica la localització del recurs dins del servidor. El camí assignat en un URL no té perquè representar un camí físic de disc amb el mateix nom o seqüència de directoris ja que els servidors web permeten fer un mapeig entre camins d'URL i directoris de disc (directoris virtuals).
- Consulta (*query*): permet passar paràmetres addicionals útils especialment quan el recurs al qual accedim és un *script* o un altre tipus d'element que executa codi en el

servidor. Està format per parells “nom=valor”, els quals, en cas d’haver-n’hi més d’un, se separen amb el caràcter &.

- Fragment: permet especificar una secció específica dins del recurs identificar per l’URL. Els navegadors no envien aquesta part de l’URL als servidors web. El navegador, un cop rep la resposta del servidor, si s’ha especificat un fragment, intenta localitzar la secció identificada pel fragment dins del contingut sencer que ha rebut i normalment el focalitza en la visualització a l’usuari.

Molts dels elements d’un URL són opcionals i es poden ometre si el context on s’utilitza l’URL permet assumir-ne uns valors per defecte. Un exemple és a la barra d’adreces d’un navegador, on no és necessari escriure “http:'/'/” a l’inici de l’URL, ja que és el valor per defecte per al navegador.

També passa en URL per fer referència a recursos o enllaços dins d’un document HTML. En aquest cas, a més, es pot ometre el nom de *host* i, fins i tot, una part del camí dels URL interns del document HTML.

Podem diferenciar entre:

URL absolut

URL que inclou el nom d’allotjador (*host*) i el camí (*path*) complet del recurs.

Alguns exemples d’URL absoluts són:

https://translate.google.es/

https://portal.edu.gva.es/aules/

URL relatiu

URL que no inclou l’esquema (*scheme*), el *host* i el port. Quan un URL és relatiu s’acostuma a anomenar camí (*path*) i es pot dividir en camins complets i camins relatius.

- Camins complets (*full paths*): comencen sempre amb el caràcter / i especifiquen tota la seqüència de directoris que cal recórrer fins arribar al recurs dins del mateix servidor d’on s’ha descarregat el document HTML, partint del directori arrel del lloc web. Per exemple, /index.html.
- Camins relatius (*relative paths*): comencen amb un caràcter diferent de / i indiquen la seqüència de directoris que cal recórrer per arribar al recurs dintre del mateix servidor d’on s’ha descarregat el document HTML, partint del directori d’on s’ha descarregat el document. Per exemple, ../aules/images/hdr_bg.gif.

Els caràcters que es poden utilitzar sense restriccions en els URL són:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 - _ . ~

També es poden usar altres caràcters, però han d'estar codificats per poder ser interpretats. Per fer-ho, s'usa la codificació percentual (*percent-encoding* o *URL encoding*):

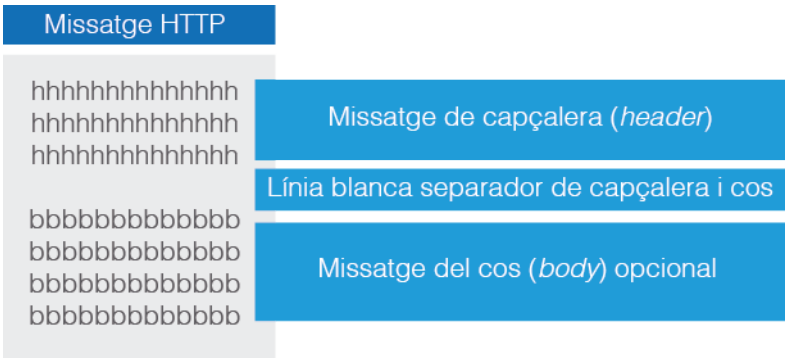
Taula: *Caràcters URL encoding*

!	#	$	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

HTTP és un protocol de petició/resposta (*request-response*). El cicle de comunicació entre el client i el servidor es basa en dos passos on la resposta per part del servidor ve precedida d'una petició per part del client.

Tot i que les peticions i les respostes contenen informació diferent, totes dues tenen una mateixa estructura formada per una capçalera i un cos. La capçalera conté informació sobre el missatge (metadades) i el cos és el que du el contingut del missatge. El contingut de les peticions i respostes pot ser buit en algunes ocasions, per tant, pot haver peticions o respostes sense contingut, només amb capçalera.

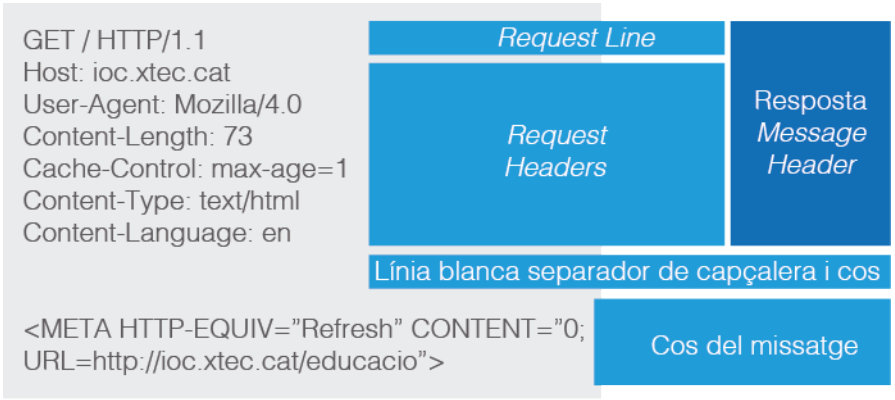
Els missatges HTTP, tant si són peticions com respostes, tenen una estructura formada per una capçalera i un cos. La capçalera està formada per informació alfanumèrica i està separada del cos per una línia en blanc (vegeu la figura). El contingut del cos (que pot ser opcional) és alfanumèric o binari. Missatge HTTP:



Les peticions HTTP, a la primera línia de la capçalera, contenen la línia de petició (*request line*). La resta de línies (opcionals) contenen les capçaleres de petició (*request headers*) tal com es veu a la figura esquema URL.

- Línia de petició: primera línia de la capçalera d'una petició. Està formada per tres camps:
  - Mètode de petició (*request method*): pot ser qualsevol dels mètodes de petició que defineix el protocol HTTP.
  - URL de petició (*request URL*): és l'URL del recurs que es demana.
  - Versió HTTP: és la versió de protocol HTTP que s'utilitzarà (HTTP/1.0 o HTTP/1.1).
- Capçalera de petició: cada capçalera va en una línia diferent i està formada per un parell "nom: valor". Si hi ha diversos valors, aquests se separen amb comes.

Figura Petició HTTP:



El protocol HTTP té definits 8 mètodes de petició (*request methods*), que són el primer que s'especifica en la línia de les peticions.

Una petició feta amb el mètode POST oculta els paràmetres, però això no implica que siga un mecanisme de seguretat. Si el protocol utilitzat és HTTP , la informació s'envia en clar a través de xarxa i pot ser interceptada durant el procés d'enviament.

1. GET: el mètode GET s'utilitza per fer la petició d'un recurs o document al servidor (un document HTML, una imatge…), especificat en el camp URL de la línia de petició (*request line*). Les peticions realitzades amb el mètode GET només haurien de fer que el servidor torne algun tipus de resposta sense que això provoque la modificació de les dades del servidor ni de l'aplicació web sobre el que s'ha executat amb el GET.

2. HEAD: el mètode HEAD s'usa per simular una petició d'un recurs al servidor. Funciona exactament igual que el GET, amb la diferència que el servidor només respon amb les capçaleres de la resposta (no envia el contingut). S'utilitza per poder fer un pronòstic de resposta d'una hipotètica petició GET. És emprat habitualment pels navegadors per comprovar les capçaleres d'un recurs determinat i així decidir si cal descarregar-lo o no en cas de tenir-ne una còpia en la memòria *cache* local (les capçaleres ens poden dir la mida del document, la data d'última modificació, etc.).

3. POST: és el mètode habitual per enviar les dades de formularis HTML que poden provocar modificacions a les dades del servidor o de l'aplicació web. Pot semblar igual que el mètode GET, però té algunes diferències importants:
   a) Els paràmetres d'un GET van concatenats a l'URL de la petició i és visible a la barra d'adreces del navegador. En canvi amb POST s'adjunten com a contingut del cos de la petició i, per tant, no són tant visibles. Com a conseqüència, les peticions GET es poden memoritzar als marcadors del navegador i es poden representar, també, com a URL en un enllaç dintre d'un document HTML. Amb POST, això no és possible.
   b) Les peticions GET es consideren idempotents (així ho defineix HTTP), la qual cosa significa que s'han de poder executar tantes vegades com es vulga sense que es produeiquen modificacions de l'estat o les dades del servidor. Per tant, els

navegadors permeten l'execució repetida de peticions GET. Per exemple, actualitzant una pàgina carregada amb GET o tirant enrere a una pàgina prèviament carregada amb GET sense donar cap avís a l'usuari. En canvi, amb el POST és a l'inrevés. Les peticions POST es consideren no idempotents (pot provocar modificacions) i els navegadors, cada cop que es pretén repetir una petició POST, informen l'usuari amb un missatge de confirmació.

4. PUT: el mètode PUT s'utilitza per poder crear o reemplaçar un determinat recurs o document del servidor. Amb el mètode PUT podem sol·licitar que el contingut present en el cos de la petició s'emmagatzeme en el servidor i passe a estar accessible a través d'un URL que indiquem a la línia de petició. Si ja existeix, se substituiria l'actual pel que donem amb el PUT. Si no existeix, se'n crearia un de nou. La utilització d'aquest mètode està restringit en la configuració per defecte dels servidors web ja que permetria la modificació no autoritzada dels seus continguts (Si ho provem, podem rebre un error del tipus "*405 Method Not Allowed*").

5. DELETE: el mètode DELETE s'utilitza per poder eliminar un determinat recurs o document del servidor. Amb el mètode DELETE es pot sol·licitar al servidor que elimine l'arxiu associat a un URL del recurs indicat a la línia de petició. La utilització d'aquest mètode està restringida en la configuració per defecte dels servidors web ja que permetria la modificació no autoritzada dels seus continguts (si ho provem, podem rebre un error del tipus "*405 Method Not Allowed*").

6. CONNECT: aquest mètode s'utilitza per poder passar connexions segures SSL a través de connexions HTTP i per poder gestionar connexions HTTP a través de servidors intermediaris (*proxies*).

7. OPTIONS: el mètode OPTIONS demana al servidor quins mètodes HTTP es poden utilitzar sobre el recurs identificat amb un URL de la línia de petició.

8. TRACE: el mètode TRACE demana al servidor que retorne una còpia de les capçaleres de la petició tal com les ha rebut. Aquest mètode HTTP sol estar desactivat per defecte.

Una vegada especificada la línia de petició d'una petició HTTP, pot haver una sèrie de capçaleres de petició (opcionals) cadascuna de les quals ocupa una línia i té el format "nom: valor".

Les capçaleres més habituals en les peticions HTTP enviades als navegadors són:

Els factors de qualitat en les capçaleres *Accept* (q) són nombres reals entre 0 i 1 que indiquen la preferència del client sobre un determinat aspecte de la resposta esperada del servidor. Per defecte, el factor de qualitat val 1.

- Allotjador (*host*): capçalera obligatòria en HTTP/1.1. El client ha d'especificar el nom del *host* al qual envia la petició (una mateixa màquina pot tenir diversos noms

de domini associats). Un mateix servidor amb una única IP pot servir diversos llocs/aplicacions web amb noms de domini diferents. Per poder identificar en quin d'ells volem enviar la petició usem aquesta capçalera.

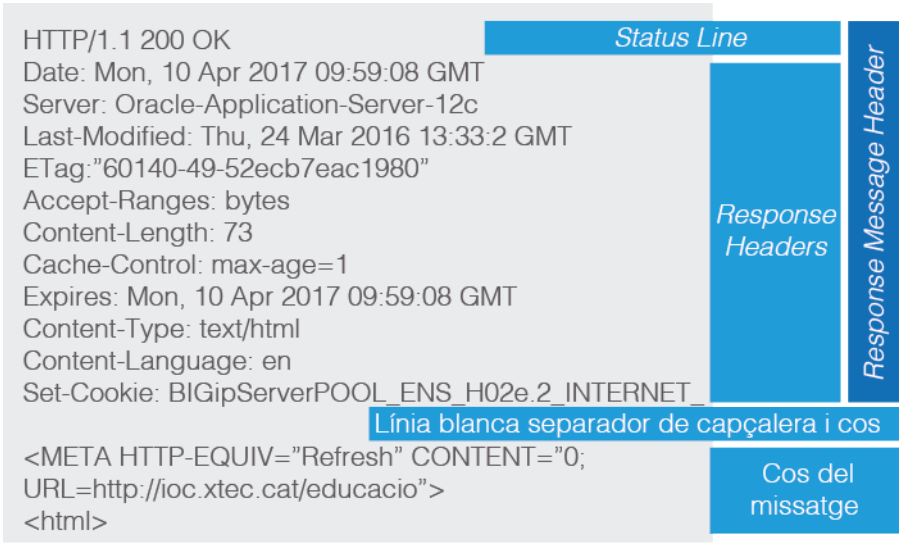
- Usuari-agent (*user-agent*): indica quin programari client (habitualment un navegador) utilitza l'usuari. Sol ser una cadena que inclou un nom identificador del navegador, un descriptor de versió i un descriptor de sistema operatiu i plataforma sobre la qual s'executa el navegador. Per exemple: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/29.0.1547.76 Safari/537.36
- *Accept*: el client pot utilitzar aquesta capçalera per indicar al servidor els tipus MIME que és capaç de gestionar i quin és el seu ordre de preferència. Si el servidor té diverses versions del recurs sol·licitat pot consultar aquesta capçalera per decidir quina s'enviarà al client. Aquest procés s'anomena negociació de tipus de continguts (*content-type negociation*). Per exemple: Accept: image/png,image/*;q=0.8,*/*;q=0.5
- *Accept-Language*: el client pot utilitzar aquesta capçalera per indicar al servidor els idiomes preferits per a les respostes obtingudes. Si el servidor té el recurs sol·licitat en diversos idiomes, pot consultar aquesta capçalera per decidir quina versió enviarà al client. Aquest procés s'anomena negociació d'idioma (*language negociation*). Per exemple: Accept-Language: ca,es
- *Accept-Charset*: el client pot utilitzar aquesta capçalera per indicar al servidor el joc de caràcters preferit per a les respostes obtingudes que incloguin informació alfanumèrica. Aquest procés s'anomena negociació de joc de caràcters (*charset negociation*). Per exemple: Accept-charset: utf-8, iso8859-1
- *Content-Type* i *content-Length*: quan la petició té un cos, és a dir, un contingut, a la capçalera s'ha d'especificar el tipus (amb un identificador MIME) i la mida (en bytes) mitjançant aquestes dues capçaleres.
- *Referer*: aquesta capçalera permet al client especificar al servidor l'adreça (URL) del recurs d'on s'ha obtingut l'URL de la petició que se li està enviant.
- *Accept-Encoding*: el client pot utilitzar aquesta capçalera per indicar al servidor els tipus de codificació que suporta. Si el servidor té el recurs comprimit o és capaç de comprimir-lo al vol en algun dels formats que accepta el client, ho pot utilitzar per reduir el temps de transmissió. El servidor ha d'indicar en la capçalera *Content-Encoding* de la resposta el tipus de compressió que ha aplicat.
- *Connection*: el client pot utilitzar aquesta capçalera per indicar al servidor si ha de tancar la connexió un cop enviada la resposta a la petició rebuda, o deixar la connexió oberta a l'espera de rebre noves peticions sense haver de repetir el procés d'establiment de la connexió. Pot tenir dos valors: *close* o *keep-alive*.

Les respostes HTTP, a la primera línia de la capçalera, contenen la línia d'estatus. La resta de línies (opcionals) contenen les capçaleres de resposta tal com es veu a la figura resposta HTTP:

- Línia d'estatus (*status line*): primera línia de la capçalera d'una resposta. Està formada per tres camps:

    - Versió HTTP: versió de protocol HTTP que utilitza el servidor (HTTP/1.0 o HTTP/1.1).
    - Codi d'estatus: codi de tres dígits que indica el resultat de la petició.
    - *Reason phrase*: xicoteta explicació del significat del codi d'estatus.
- Capçalera de resposta (*response header*): cada una va en una línia diferent i està formada per un parell "nom: valor". Si hi ha diversos valors, se separen amb comes.

Resposta HTTP:



El protocol HTTP defineix 5 codis d'estat (*status codes*) que permeten indicar diversos tipus de situacions que es poden donar com a resposta a una petició d'un client.

- 1xx: són de tipus informatiu, informen el client que la petició ha estat rebuda i que el servidor continua processant la resposta.
- 2xx: indiquen la petició ha estat correcta i s'ha processat satisfactòriament.
- 3xx: indiquen alguna forma de redirecció. Amb un codi d'aquesta sèrie es dona a entendre al client que la petició és correcta però que la resposta s'ha d'obtenir d'algun altre lloc.
- 4xx: indiquen que hi ha hagut una errada en el processament de la petició perquè el client ha fet alguna cosa malament, l'error ha estat causat pel client.
- 5xx: indiquen que hi ha hagut una errada en el processament de la petició a causa d'una fallada en el servidor, l'error ha estat causat pel servidor.

Un cop especificada la línea d'estat d'una resposta HTTP, hi ha una sèrie de capçaleres de resposta (opcionals) cadascuna de les quals ocupa una línia i té el format "nom: valor". Les capçaleres més habituals en les respostes HTTP són:

- *Content-Base*: capçalera que conté un URL per ser utilitzat com a base per als URL relatius que hi hagi als documents HTML que s'enviïn amb la resposta.
- *Content-Length*: capçalera de resposta que indica la mida en bytes del cos associat a la resposta.

- *Content-Type*: capçalera que indica el tipus de contingut del cos associat a la resposta. Els *media types* reconeguts estan definits per la Internet Assigned Numbers Authority (IANA).
- *Date*: aquesta capçalera, obligatòria en totes les respostes en HTTP/1.1, indica la data i hora d'enviament de la resposta, és a dir, quan surt del servidor.
- *Last-Modified*: capçalera que indica la data i hora en la qual el recurs demanat ha estat actualitzat per últim cop. L'objectiu d'aquesta capçalera és permetre la gestió de *cache* de recursos. No funciona correctament per a recursos dinàmics. Per solucionar-ho es va introduir la capçalera ETag.
- *ETag*: permet que el servidor puga enviar un *hash* o *checksum* del cos de la resposta etiqueta d'entitat (*entity tag*) per ser utilitzat per al control de *cache* de recursos HTTP en els clients i servidors proxies. Es pot considerar que totes les còpies d'un recurs amb el mateix URL i la mateixa etiqueta d'entitat són idèntics. Per tant, si se'n té una còpia en memòria *cache* no cal descarregar-los un altre cop.
- *Server*: és l'equivalent a la capçalera de petició *User-Agent*, però del costat del servidor. Serveix perquè el servidor web puga indicar el seu nom i versió.
- *Location*: indica al client cap on ha d'anar a buscar un recurs que hagi demanat i que no es troba a l'URL de la petició (resposta amb un codi d'estat de la sèrie 3xx).

3. Configuració avançada del servidor web

En aquesta unitat fareu servir el servidor HTTP anomenat Apache. Apache és un servidor HTTP (de pàgines web) de codi obert multiplataforma desenvolupat per Apache Software Foundation. Apache presenta, entre altres característiques, missatges d'error altament configurables, bases de dades d'autenticació i negociació de continguts, però s'ha criticat per la manca d'una interfície gràfica que ajude a configurar-lo.

Configuració d'Apache

El sistema on realitzareu la instal·lació i configuració està basat en un sistema Debian (Ubuntu o semblants).

El conjunt d'instruccions d'instal·lació o configuració mitjançant ordres de sistema és dependent del sistema operatiu. Ssi aneu a un sistema REDHAT, CENTOS, etc. heu de mirar el procediment d'instal·lació i les ordres específiques del sistema.

La configuració interna és independent del sistema operatiu que fem servir, qualsevol configuració que realitzeu durant aquests apartats és compatible amb altres sistemes operatius basats en Unix.

Per iniciar el procés d'instal·lació d'Apache, primer de tot obriu un terminal i executeu les ordres d'actualització dels repositoris d'Ubuntu:

```
sudo apt update
```

```
root@2f24df58d1b8:/# sudo apt update
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
29 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@2f24df58d1b8:/#
```

sudo apt install apache2

```
root@2f24df58d1b8:/# sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils bzip2 ca-certificates file
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  libbrotli1 libcurl4 libexpat1 libgdbm-compat4 libgdbm6 libicu70
  libjansson4 libldap-2.5-0 libldap-common liblua5.3-0 libmagic-mgc
  libmagic1 libnghttp2-14 libperl5.34 libpsl5 librtmp1 libsasl2-2
  libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh-4 libxml2
  mailcap media-types mime-support netbase openssl perl perl-base
  perl-modules-5.34 publicsuffix ssl-cert xz-utils
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
  www-browser ufw bzip2-doc gdbm-l10n libsasl2-modules-gssapi-mit
  | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap
  libsasl2-modules-otp libsasl2-modules-sql perl-doc
  libterm-readline-gnu-perl | libterm-readline-perl-perl make
  libtap-harness-archive-perl
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils bzip2 ca-certificates
  file libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  libbrotli1 libcurl4 libexpat1 libgdbm-compat4 libgdbm6 libicu70
  libjansson4 libldap-2.5-0 libldap-common liblua5.3-0 libmagic-mgc
  libmagic1 libnghttp2-14 libperl5.34 libpsl5 librtmp1 libsasl2-2
  libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh-4 libxml2
  mailcap media-types mime-support netbase openssl perl
  perl-modules-5.34 publicsuffix ssl-cert xz-utils
The following packages will be upgraded:
  perl-base
1 upgraded, 43 newly installed, 0 to remove and 28 not upgraded.
Need to get 27.4 MB of archives.
After this operation, 111 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Una vegada accepteu la instal·lació, el procés APT instal·la Apache.

Una vegada finalitzada la instal·lació procediu a verificar el resultat d'executar l'ordre següent:

```
sudo service apache2 status
```

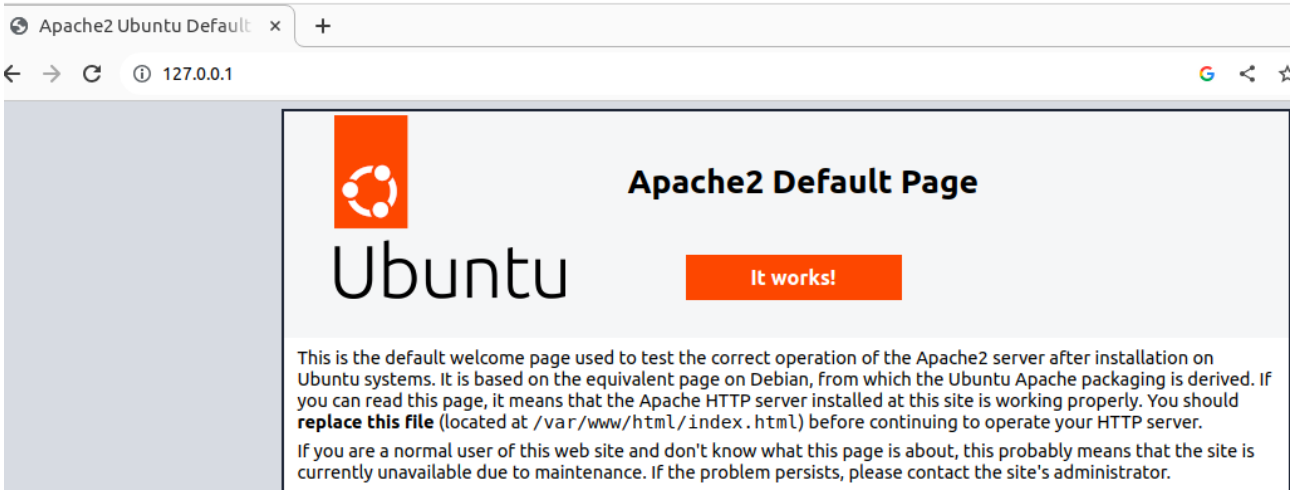
```
root@2f24df58d1b8:/# sudo service apache2 status
 * apache2 is running
root@2f24df58d1b8:/#
```

Com podeu veure, indica que el servidor està en funcionament.

A continuació, des d'un navegador, executeu:

localhost

Si tot ha anat bé ha de mostrar una pàgina web per informar de la correcta instal·lació del servidor Apache, tal com podeu veure:



Una vegada verificat que Apache funciona, la configuració del servidor es fa mitjançant un fitxer de text pla, de la mateixa manera que se sol fer amb molts serveis d'Unix.

El directori on hi ha les configuracions és dins */etc/apache2*, i el fitxer de configuració principal és el fitxer *apache2.conf*.

Editar el fitxer original pot comportar modificar el fitxer i perdre les dades inicials de configuració. Sempre és bo fer una còpia de seguretat del fitxer amb l'ordre cp (per copiar fitxers).

Navegueu fins al directori de configuracions d'Apache */etc/apache2* i executeu:

```
cd etcapache2
ls -l
```

Amb aquesta ordre es llista tot el contingut del directori, amb el resultat següent:

Com

```
root@2f24df58d1b8:/# cd /etc/apache2
root@2f24df58d1b8:/etc/apache2# ls -l
total 80
-rw-r--r-- 1 root root  7224 May  3 20:02 apache2.conf
drwxr-xr-x 2 root root  4096 Oct 28 09:56 conf-available
drwxr-xr-x 2 root root  4096 Oct 28 09:56 conf-enabled
-rw-r--r-- 1 root root  1782 May  3 20:02 envvars
-rw-r--r-- 1 root root 31063 May  3 20:02 magic
drwxr-xr-x 2 root root 12288 Oct 28 09:56 mods-available
drwxr-xr-x 2 root root  4096 Oct 28 09:56 mods-enabled
-rw-r--r-- 1 root root   320 May  3 20:02 ports.conf
drwxr-xr-x 2 root root  4096 Oct 28 09:56 sites-available
drwxr-xr-x 2 root root  4096 Oct 28 09:56 sites-enabled
root@2f24df58d1b8:/etc/apache2#
```

podeu veure, el directori */etc/apache* conté diferents fitxers, incloent apache2.conf. Els fitxers que hi ha dins fan referència a fitxers de configuracions de modularitats que amplien la capacitat d'Apache, com pot ser: afegir mòduls com PHP, configurar la seguretat per xifrar la informació, crear servidors virtuals, etc.

Editeu el fitxer apache2.conf.

```
sudo nano apache2.conf
```

Caràcter #: Abans de detallar configuracions, fixeu-vos en el fitxer apache2.conf. Moltes de les línies del fitxer de configuració estan iniciades amb el caràcter # (*hashtag*).

El caràcter # es fa servir en múltiples configuracions de serveis del sistema. Permet comentar línies de configuració i fer comentaris, ja que l'intèrpret de l'aplicació no els executa.

Com podeu comprovar, el fitxer de configuració *apache2.conf* és bastant extens. Tot seguit es detallen les opcions de configuració més rellevants.

apache2.conf

Apache té una gran extensió de directives per configurar el servidor HTTP. Per la seva extensió natural no es pot detallar tot. Per a més detall adreceu-vos a la documentació oficial.

Dins de la configuració inicial, es poden trobar algunes de les directives més importants:

- ***Timeout***: nombre de segons abans que rep i envia el temps d'espera.
- ***Keep-alive***: permet acceptar connexions persistents.
- ***MaxKeepAliveRequests***: nombre màxim de sol·licituds de permís durant una connexió persistent.
- ***KeepAliveTimeout***: nombre de segons d'espera per a la següent petició del mateix client en la mateixa connexió.
- ***ErrorLog***: ubicació de l'arxiu de registre d'errors.
- ***Include module configuration***: camí on es troba el fitxer de mòduls disponibles i actius.

- ***Include list of ports to listen***: camí on es troba el fitxer dels ports d'escolta del servidor.
- ***DocumentRoot***: estableix el directori de publicació del seu web principal o per defecte.
- ***Directory***: per a cada directori que calga configurar es pot definir un bloc d'opcions de configuració agrupades en aquesta directiva.
- ***DirectoryIndex***: documents que es mostren per defecte quan se sol·licita un URL i no s'especifica el document.
- ***AccessFileName***: nom de l'arxiu que ha de buscar en cada directori per a directives de configuració addicionals.
- ***Include the virtual host configurations***: camí on es troba el fitxer de configuració dels servidors virtuals.
- ***User***: compte d'usuari que el servidor web Apache utilitzarà mentre s'execute (determina els permisos d'accés que tindrà sobre directoris i arxius).
- ***Group***: grup d'usuaris que el servidor web Apache utilitzarà mentre s'executa (igual que amb l'usuari, determina els permisos d'accés que tindrà sobre directoris i arxius).

Amb les opcions d'instal·lació per defecte, ja tindreu el lloc web per defecte a la carpeta:

/var/www

Aquest lloc web, per defecte, és l'ofert per a qualsevol connexió que es faça sobre el port 80 en qualsevol de les adreces IP o noms de domini establerts sobre el servidor. Per modificar el port cal modificar el fitxer ***ports.conf***.

4. Mòduls: instal·lació, configuració i ús

La gran majoria de servidors web permeten la instal·lació de mòduls per ampliar les seues funcionalitats. Tenir funcionalitats en forma de mòdul permet adaptar millor el consum de recursos del servidor web a les nostres necessitats de producció (el servidor web només carregarà i executarà els mòduls que com a administradors tenim instal·lats i configurats).

El servidor web Apache HTTP Server permet afegir funcionalitats addicionals a les que ens ofereix la seua configuració bàsica en forma de mòduls instal·lables.

Instal·lació de mòduls en Apache HTTP Server

Treballant en una plataforma amb un sistema Debian, la instal·lació de mòduls es fa normalment mitjançant el gestor d'instal·lació de paquets APT. La instal·lació d'un mòdul afegeix els arxius següents en el sistema:

- El fitxer del mòdul (habitualment el nom sol començar per *mod* i acabar en *.so*) es guarda al directori */usr/lib/apache2/modules*. Si feu un *ls –l* de */usr/lib/apache2/modules* podeu veure el contingut de la carpeta similiar a:

```
root@08ad07347c97:/lib/apache2/modules# ls -l
total 4320
```

```
-rw-r--r-- 1 root root  17172 May  3 20:02 httpd.exp
-rw-r--r-- 1 root root  14704 May  3 20:02 mod_access_compat.so
-rw-r--r-- 1 root root  14704 May  3 20:02 mod_actions.so
-rw-r--r-- 1 root root  18800 May  3 20:02 mod_alias.so
-rw-r--r-- 1 root root  14704 May  3 20:02 mod_allowmethods.so
-rw-r--r-- 1 root root  14624 May  3 20:02 mod_asis.so
-rw-r--r-- 1 root root  18800 May  3 20:02 mod_auth_basic.so
-rw-r--r-- 1 root root  39280 May  3 20:02 mod_auth_digest.so
(i continua...)
```

- El fitxer que conté la directiva per carregar del mòdul (LoadModule) té com a extensió *.load*, i es guarda al directori */etc/apache2/mods-available*.
- El fitxer que conté les directives de configuració del mòdul (en cas que en tinga), amb una configuració per defecte, acostuma a tenir una extensió *.conf*, i està emmagatzemat al directori */etc/apache2/mods-available*. No obstant això, no tots els mòduls venen amb el corresponent fitxer de directives de configuració.

Activació i desactivació de mòduls en Apache HTTP Server

Una vegada instal·lats els mòduls, s'han d'activar. Amb servidors Apache es poden tenir mòduls instal·lats però no activats, és a dir, que no s'estan utilitzant.

Per activar un mòdul cal crear un enllaç simbòlic dins del directori */etc/apache2/mods-enabled* sobre el fitxer *.load* i sobre el fitxer *.conf* (si té fitxer associat) que conté la directiva de càrrega del mòdul i les directives de configuració respectivament. Aquests fitxers són */etc/apache2/mods-available*. Per facilitar la configuració i l'administració del servidor s'acostuma a posar el mateix nom del fitxer a l'enllaç simbòlic.

També es poden activar els mòduls usant les eines que faciliten el servidor Apache i el sistema operatiu. Disposeu de l'ordre *a2enmod*, que permet activar un mòdul sempre que estiga instal·lat. Si li doneu com a paràmetre el nom del mòdul que s'ha d'activar, crea l'enllaç o enllaços simbòlics necessaris dins el directori */etc/apache2/mods-enabled*.

Per exemple:

```
a2enmod rewrite
```

```
root@08ad07347c97:/lib/apache2/modules# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  service apache2 restart
root@08ad07347c97:/lib/apache2/modules#
```

Un cop instal·lats i activats els mòduls, ja poden ser utilitzats. A partir d'ací, podeu canviar la configuració manipulant el fitxer *.conf* que estiga associat al mòdul. Si no existeix aquest fitxer, pot ser perquè el mòdul no té directives de configuració o perquè per defecte no porta el fitxer (hi ha la possibilitat que es puga crear i posar-hi algunes directives). Per poder canviar la configuració d'un mòdul, heu de consultar la

documentació tècnica per saber quines directives de configuració ofereix i quines opcions posibles hi ha per a cada directiva.

Per desactivar un mòdul hi ha dues opcions, igual que abans. O bé esborreu els enllaços simbòlics que heu creat o bé useu l'ordre *a2dismod*.

Per exemple:

`a2dismod rewrite`

```
root@08ad07347c97:/lib/apache2/modules# a2dismod rewrite
Module rewrite disabled.
To activate the new configuration, you need to run:
  service apache2 restart
root@08ad07347c97:/lib/apache2/modules#
```

Els mòduls d'Apache són opcionals (no tenen perquè estar carregats), les seues directives de configuració d'un mòdul en concret poden provocar errors en cas que el mòdul no estiga activat. Per evitar això, Apache disposa del bloc *<IfModule>*, que es pot incloure en els seus fitxers de configuració que permeten indicar que unes determinades directives de configuració tinguen efecte sobre el servidor només si el mòdul que s'indica està actiu.

```
<IfModule nom_mòdul.c>
  # Directives de configuració del mòdul, en cas que estigui activat.
  ...
</IfModule>
```

Per saber quins mòduls estan actius, podeu llistar el contigut del directori amb

`ls -l /etc/apache2/mods-enabled`

```
root@08ad07347c97:/lib/apache2/modules# ls -l /etc/apache2/mods-enabled
total 0
lrwxrwxrwx 1 root root 36 Oct 28 09:56 access_compat.load -> ../mods-available/access_compat.load
lrwxrwxrwx 1 root root 28 Oct 28 09:56 alias.conf -> ../mods-available/alias.conf
lrwxrwxrwx 1 root root 28 Oct 28 09:56 alias.load -> ../mods-available/alias.load
lrwxrwxrwx 1 root root 33 Oct 28 09:56 auth_basic.load -> ../mods-available/auth_basic.load
lrwxrwxrwx 1 root root 33 Oct 28 09:56 authn_core.load -> ../mods-available/authn_core.load
lrwxrwxrwx 1 root root 33 Oct 28 09:56 authn_file.load -> ../mods-available/authn_file.load
lrwxrwxrwx 1 root root 33 Oct 28 09:56 authz_core.load -> ../mods-available/authz_core.load
lrwxrwxrwx 1 root root 33 Oct 28 09:56 authz_host.load -> ../mods-available/authz_host.load
lrwxrwxrwx 1 root root 33 Oct 28 09:56 authz_user.load -> ../mods-available/authz_user.load
lrwxrwxrwx 1 root root 32 Oct 28 09:56 autoindex.conf -> ../mods-available/autoindex.conf
lrwxrwxrwx 1 root root 32 Oct 28 09:56 autoindex.load -> ../mods-available/autoindex.load
lrwxrwxrwx 1 root root 30 Oct 28 09:56 deflate.conf -> ../mods-available/deflate.conf
lrwxrwxrwx 1 root root 30 Oct 28 09:56 deflate.load -> ../mods-available/deflate.load
lrwxrwxrwx 1 root root 26 Oct 28 09:56 dir.conf -> ../mods-available/dir.conf
lrwxrwxrwx 1 root root 26 Oct 28 09:56 dir.load -> ../mods-available/dir.load
lrwxrwxrwx 1 root root 26 Oct 28 09:56 env.load -> ../mods-available/env.load
lrwxrwxrwx 1 root root 29 Oct 28 09:56 filter.load -> ../mods-available/filter.load
lrwxrwxrwx 1 root root 27 Oct 28 09:56 mime.conf -> ../mods-available/mime.conf
```

L'ordre `apache2ctl –l` llista els mòduls compilats que el servidor Apache porta integrats (mòduls que no es poden descarregar o desactivar).

```
root@08ad07347c97:/lib/apache2/modules# apache2ctl -l
Compiled in modules:
  core.c
  mod_so.c
  mod_watchdog.c
  http_core.c
  mod_log_config.c
  mod_logio.c
  mod_version.c
  mod_unixd.c
root@08ad07347c97:/lib/apache2/modules#
```

Per saber quins mòduls hi ha disponibles per instal·lar al servidor, amb el gestor de paquets APT podeu executar la comanda *apt-cache search libapache2-mod*. Es mostrarà un llistat amb el nom del mòdul i una xicoteta descripció de cadascun.

```
root@08ad07347c97:/lib/apache2/modules# apt-cache search libapache2-mod
libapache2-mod-apparmor - changehat AppArmor library as an Apache module
libapache2-mod-auth-mellon - SAML 2.0 authentication module for Apache
libapache2-mod-auth-pgsql - Module for Apache2 which provides PostgreSQL authentication
libapache2-mod-auth-plain - Module for Apache2 which provides plaintext authentication
libapache2-mod-perl2 - Integration of perl with the Apache2 web server
libapache2-mod-perl2-dev - Integration of perl with the Apache2 web server - development files
libapache2-mod-perl2-doc - Integration of perl with the Apache2 web server - documentation
libapache2-mod-php - server-side, HTML-embedded scripting language (Apache 2 module) (default)
libapache2-mod-php8.1 - server-side, HTML-embedded scripting language (Apache 2 module)
libapache2-mod-wsgi-py3 - Python 3 WSGI adapter module for Apache
libapache-mod-jk-doc - Documentation of libapache2-mod-jk package
libapache2-mod-apreq2 - generic Apache request library - Apache module
libapache2-mod-auth-cas - CAS authentication module for Apache2
```

(i continua...)

5. Servidors virtuals. Creació, configuració i utilització

El protocol DNS permet tenir diversos noms de domini que apunten sobre un mateix servidor (pot ser una mateixa IP o poden ser adreces IP diferents que corresponguen a un mateix servidor).

Mitjançant el protocol DNS es pot aconseguir que un servidor ofereisca llocs webs diferents en funció del nom de domini que s'haja utilitzat per establir la connexió, gràcies, també, a la capçalera *host* que proporciona el protocol HTTP. Això és el que es coneix com a servidors virtuals (un mateix servidor que actua com si foren diversos servidors web).

En la terminologia d'Apache s'anomena *virtual host* o *vhost* cada un dels servidors virtuals que hi ha en funcionament a banda del servidor principal o per defecte:

- Quan s'assignen servidors virtuals diferents a adreces IP diferents es parla de **servidors virtuals basats en IP** o *IP-based vhosts*.
- Quan s'assignen múltiples seus virtuals a una mateixa adreça IP es parla de **servidors virtuals basats en nom** o *Name-based vhosts*.

ELs passos que s'han de seguir per crear i posar en marxa un servidor virtual són:

Tot procediment de configuració s'ha de dur a terme amb privilegis de superusuari (*root*).

1. Aneu al directori */etc/apache2/sites-available*. Aquesta carpeta conté els fitxers de configuració per a cadascun dels servidors virtuals disponibles al servidor.
2. Creeu un nou fitxer de configuració del lloc web amb el nom que vulgueu, dins del mateix directori.
3. Afegiu les opcions per adaptar-lo a les necessitats del nou servidor virtual dintre d'aquest fitxer.
4. Creeu l'enllaç simbòlic dintre la carpeta */etc/apache2/sites-enabled* que apunte cap al fitxer de configuració del servidor virtual creat a la carpeta */etc/apache2/sites-available*. O bé useu l'ordre *a2ensite nom_fitxer_servidor_virtual.conf*.
5. Executeu l'ordre *service apache2 reload*. Aquesta ordre força al servidor Apache a tornar a carregar la seva configuració.
6. Comproveu com respon el servidor facilitant el lloc web corresponent a cada servidor virtual.

Vegeu un exemple de configuració d'un servidor virtual:

```
<VirtualHost *:80>

# Aquest servidor virtual serà el que actuarà quan a la capçalera Host d'una
# petició HTTP hi trobem "www.dawcullera.org".

 ServerName  www.dawcullera.org

# Directori arrel del lloc web (directori físic de disc).

 DocumentRoot var/www/html/dawcullera

# Opcions de configuració per al directori arrel del lloc web.
 <Directory /webs/enciams>
   Options –FollowSymLinks –Indexes +MultiViews AllowOverride None
   Order allow,deny
   Allow from all
 </Directory>

# Opcions de configuració dels logs del servidor virtual.
 ErrorLog ${APACHE_LOG_DIR}/daw.error.log
 LogLevel warn
 CustomLog ${APACHE_LOG_DIR}/daw.access.log combined

</VirtualHost>
```

Vegeu l'anàlisi detallada de les principals opcions de configuració necessàries per definir una seu virtual:

- **VirtualHost**: aquesta directiva és la que fa el lligam amb l'adreça IP i port assignats a la seu virtual. Tot i que, per claredat, en l'exemple s'ha indicat un nom d'amfitrió (*host*) en lloc d'una adreça IP, Apache recomana usar sempre l'adreça IP.
- **ServerAdmin**: indica el nom de l'administrador de la seu virtual. De fet, n'indica el correu electrònic.
- **DocumentRoot**: defineix el directori de publicació de la seu web virtual. El directori que s'indica és una ruta absoluta del sistema físic de fitxers, no una ruta relativa del servidor web.

- **ServerName**: és el nom virtual amb el qual es reconeix aquest web, el nom que els clients han de referenciar per poder accedir al web.
- **errorLog** i **CustomLog**: aquestes dues directives especifiquen la ubicació dels fitxers de registre o logs de monitoratge de l'activitat d'aquesta seu web. Les rutes que s'hi indiquen són relatives i s'utilitza el directori de logs definit en la configuració global.

Cal tenir presents algunes qüestions relatives als permisos d'accés a carpetes i fitxers:

Al directori */etc/apache2* hi ha un fitxer anomenat *envvars* on, entre d'altres aspectes, es defineix el nom d'usuari i el grup amb el qual treballarà el servidor Apache.

Per defecte trobarem:

export APACHE_RUN_USER=www-data

export APACHE_RUN_GROUP=www-data

Això indica que el servidor Apache accedirà a arxius i directoris utilitzant l'usuari *www-data* i el grup *www-data*. Tenint en compte aquest tema, caldrà donar permís de lectura als fitxers (r–) i accés als directoris (r-x), com a mínim, a aquest usuari/grup a tots els directoris i fitxers que formen part dels llocs web que serveisca l'Apache.

6. Autenticació i control d'accés

Fins ara heu vist com crear i configurar diverses seus web accessibles per a tots els que tinguen accés al servidor. Hi ha ocasions en què es vol restringir l'accés a una part del web o a tot el web però només per a uns usuaris concrets. En aquest apartat es descriuen diverses formes de fer-ho.

El servei web incorpora mecanismes bàsics per verificar els usuaris que volen accedir a àrees restringides. Però, a més a més, la flexibilitat dels mòduls fa que es puguen afegir nous mecanismes que més avant tot i que no hagen estat desenvolupats per Apache. Així, per validar l'accés a un directori amb material dels professors en un web d'una escola segurament n'hi ha prou amb el mecanisme bàsic de verificació d'usuaris i grups. En canvi, per accedir a un web ultrasecret d'una agència governamental potser cal incorporar mecanismes addicionals, basats per exemple en l'empremta òptica i el registre de veu.

Primerament cal analitzar els mecanismes de validació d'usuaris generals que permet el servidor web:

- **Autenticació bàsica amb fitxers**: el mecanisme més simple per implementar el control d'accés a recursos d'una seu web és utilitzar fitxers d'**usuaris** i **grups** propis del servidor. Apache proporciona eines per crear-los. L'avantatge principal d'aquest mètode és la facilitat d'administració. L'inconvenient és que comporta una gestió diferenciada dels usuaris del servei web i dels del sistema. De fet, això pot ser un inconvenient o un avantatge, si el que interessa és tenir-los segregats.
- **Autenticació mitjançant PAM**: en els sistemes GNU/Linux actuals l'autenticació dels usuaris es fa via PAM (*Pluggable Authentication Module*). El PAM comprova el directori /etc/passwd, el LDAP, el Kerberos, les empremtes dactilars o el que calga.

Usar el lligam amb el mòdul del PAM és un bon mecanisme per validar els usuaris del servei web igual que es validen els usuaris del sistema.

- **Autenticació mitjançant LDAP**: un dels mecanismes més populars actualment per a l'autenticació és el LDAP. El mòdul del LDAP permet passar la validació dels usuaris a l'encarregat de gestionar l'autenticació LDAP dels usuaris del sistema. També es pot tenir en funcionament un servei LDAP específic per a les validacions del servei web.

Alguns conceptes clau relacionats amb el control d'accés al servidor són:

- **Autenticació**: el procés d'autenticació és el que determina si un usuari és qui diu ser. En cap moment governa quins drets té, què pot fer i què no, simplement s'encarrega de comprovar que l'usuari és qui diu que és. Per implementar l'autenticació hi ha innumerables sistemes, des dels fitxers d'usuaris i contrasenyes fins a sofisticats mecanismes d'empremtes dactilars, òptiques, dades biomètriques o llapis USB (sense el llapis l'usuari no es pot identificar).
- **Autorització**: un cop s'ha identificat un usuari (és qui diu ser), què pot fer?, a quins recursos pot accedir?, a quins no? Això és l'autorització: determinar els drets d'utilització dels recursos.
    - **Recurs amb accés restringit**: el control d'accés al servidor busca determinar quins recursos són accessibles per a quins usuaris. Pot restringir l'accés a tota una seu web de manera que només els usuaris autoritzats puguen accedir als seus continguts. Sovint es restringeixen àrees concretes de la seu web, per exemple directoris que són accessibles només per a un conjunt d'usuaris (els empleats, o els professors, en el web de l'escola). En aquest cas parlem de directoris amb accés restringit.
- **Reialme**: en una seu web hi poden haver diverses àrees restringides a perfils d'usuari diferents. Els reialmes permeten definir quines àrees restringides comparteixen el mateix grau d'accés.

Exemple de seu web d'una escola

Tornem a l'exemple d'una seu web d'una escola on hi ha tot de continguts públics accessibles per a tots. El directori Notes és un recurs restringit on només hi poden accedir els alumnes de l'escola. Els directoris Programacions i Registres de Treball són accessibles només per als professors. Un professor que, per exemple, s'autentica per entrar a l'àrea Programacions introduint el seu identificador d'usuari i contrasenya, si vol entrar a l'àrea Registres de Treball s'hauria de tornar a identificar entrant de nou l'usuari i la contrasenya. Els reialmes permeten declarar que diversos llocs restringits tenen el mateix nivell d'accés, de manera que si un usuari s'ha autenticat en un està autenticat en tots els recursos que formen part del reialme.

- **Web amb inici de nom d'usuari/contrasenya**: un error molt típic és confondre l'autenticació de servidor amb l'autenticació de programari que fan les seus webs. Quan un usuari es valida en un entorn web com per exemple Yahoo o Google, no està usant l'autenticació amb el servidor web. Està usant un usuari i una contrasenya de l'empresa web a la qual es connecta i la gestió d'aquesta sessió d'usuari per consultar el seu correu es fa mitjançant la programació en les mateixes

pàgines web que visita. Això no té res a veure amb el control d'accés al servidor que es tracta en aquest apartat.

Autenticar els usuaris és determinar de forma veraç si un usuari és qui diu ser. **Autoritzar** és indicar quins usuaris tenen dret a accedir a quins recursos. Les seus web i els directoris que limiten l'accés a un conjunt restringit d'usuaris s'anomenen **recursos restringits**. Els recursos restringits que implementen la mateixa política de seguretat es poden agrupar en reialmes.

Els mòduls de control d'accés

Apache gestiona l'autenticació i el control d'accés al servidor mitjançant mòduls propis (i també es poden incorporar mòduls externs). Cada mòdul consta d'un conjunt de directives que permeten configurar el funcionament de l'autenticació i control d'accés implementats. Aquests mòduls es poden classificar en tres categories segons la seva funcionalitat:

1) Tipus d'autenticació: l'autenticació pot ser de tipus *basic* o *digest*. En aquests exemples s'utilitza autenticació bàsica. L'autenticació *digest* implica comunicacions xifrades. Aquests mòduls s'implementen amb la directiva *AuthType*.

AuthType Basic

Podeu observar que els mòduls identifiquen en el seu nom la cadena **auth** d'*authentication*.

mod_auth_basic
mod_auth_digest

2) Proveïdor d'autenticació: indica quin és el mecanisme usat per realitzar l'autenticació. Són els mòduls que permeten autenticar usant fitxers de contrasenyes o el mòdul PAM o el de LDAP. Es poden identificar els mòduls d'aquesta família perquè inclouen en el seu nom la cadena **authn** d'*authentication*.

mod_authn_file
mod_authn_alias
mod_authnz_ldap
...

3) Autorització: els mòduls d'aquesta família proporcionen autorització d'usuari, de grups, del LDAP o del que convinga. Aquests mòduls es determinen segons el valor que prenga la directiva ***require***.

Require user valid–user

Podeu observar que els mòduls identifiquen en el seu nom la cadena **authz** d'*authorization*.

mod_authz_user
mod_authz_group
mod_authz_owner

mod_authz_ldap
...

Autenticació bàsica amb fitxers

El mecanisme més senzill per implementar l'autenticació en el servidor és l'autenticació bàsica amb fitxers d'usuaris i grups específics per al servidor web. Això es pot interpretar com un desavantatge perquè obliga a portar una gestió d'usuaris a més de la gestió d'usuaris del sistema. Però al mateix temps és un avantatge si el que volem és segregar aquets dos conjunts d'usuaris i administrar-los per separat.

Amb l'autenticació bàsica utilitzant fitxers es poden validar els usuaris utilitzant un **fitxer d'usuaris**, que conté els comptes d'usuaris i les seves contrasenyes.

També es poden validar grups d'usuaris amb un **fitxer de grups**, que indica quins usuaris formen part de cada grup.

El procés més simplificat per implementar la verificació d'usuaris i grups mitjançant fitxers de text pla amb contrasenyes requereix els passos següents:

1. Crear el fitxer d'usuaris en què s'indica la contrasenya corresponent a cada usuari.
2. Crear el fitxer de grups assignant a cada grup els usuaris que en formen part.
3. Identificar (o crear) el recurs que ha de tenir l'accés restringit.
4. Definir les directives apropiades per restringir l'accés al recurs no-més als usuaris autoritzats.

L'exemple següent crearà un directori anomenat *m08* en el web www.culleradaw.org, al qual només podran accedir els usuaris autoritzats (els professors). Primer cal crear el fitxer d'usuaris. Es tracta d'un fitxer de text pla en el qual s'emmagatzemen l'identificador i la contrasenya, que pot ser en text pla o xifrada, de cada usuari. Per crear el fitxer i cada nou usuari s'utilitza l'ordre ***htpasswd***, proporcionada pel paquet del servidor. En el primer exemple s'utilitza l'opció *-c*, que crea el fitxer de nou.

```
root@a7b4cf9809a6:/var/www# htpasswd -c passwd/passwd alumne1
New password:
Re-type new password:
Adding password for user alumne1
root@a7b4cf9809a6:/var/www# htpasswd  passwd/passwd alumne2
New password:
Re-type new password:
Adding password for user alumne2
root@a7b4cf9809a6:/var/www#
root@a7b4cf9809a6:/var/www# cat passwd/passwd
alumne1:$apr1$ckTPTRNy$Dj3JHc/NDRv1Cx.EWIKkS.
alumne2:$apr1$8juUK1Zz$skIGDN3YMYPOhjDGRRDIr.
root@a7b4cf9809a6:/var/www#
```

A continuació cal posar en cada grup (de moment no n'hi ha cap) els usuaris que n'han de formar part. De fet, és tan senzill com crear un fitxer de text pla cada línia del qual consta del nom del grup, el delimitador dos punts (:) i la llista d'usuaris separats per espais.

```
root@a7b4cf9809a6:/var/www/passwd# cat group
alumnes:  alumne1 alumne2
profes: professor
root@a7b4cf9809a6:/var/www/passwd#
```

Ara cal generar el **recurs restringit**, l'accés al qual només es permetrà als usuaris autoritzats. En aquest cas serà un directori anomenat *daw* en el web www.dawcullera.org

root@a7b4cf9809a6:/var/www# mkdir daw
root@a7b4cf9809a6:/var/www# nano daw/index.html
... crear una pàgina ....

Finalment s'ha d'assignar al directori local les directives apropiades per convertir-lo en un recurs d'accés restringit. Caldrà modificar el fitxer de configuració global httpd.conf i definir un bloc de configuració usant la directiva ***Directory***. En aquesta directiva cal indicar la ruta absoluta corresponent al sistema de fitxers real del servidor (no és possible usar rutes relatives al servei web).

<Directori path–absolut–filesystem>
... opcions de configuració ...
</Directory>

Un exemple complet de configuració és el que es mostra a continuació, en el qual únicament es permet accedir al recurs a usuaris del grup dels professors anomenat *profes*:

```
<Directori /var/www/daw>
 AuthType Basic
 AuthName "Restringit a professors"
 AuthBasicProvider file
 AuthUserFile /var/www/passwd/passwd
 AuthGroupFile /var/www/passwd/group
 Require group profes
</Directory>
```

Repassem les directives que s'hi utilitzen:

AuthType: indica que el tipus d'autenticació és bàsica (en lloc de *digest*).

AythName: declara el reialme al qual pertany el recurs restringit. Això permet que si hi ha altres recursos restringits associats a aquest reialme l'usuari que ja s'ha autenticat en un d'ells no ho haja de fer en els altres. El nom del reialme el posa l'administrador web.

AuthBasicProvider: indica el mètode d'autenticació que s'ha d'usar. Pot prendre valors tipus *ldap*, *pam*, *dbm*, *bdb*, *file* i d'altres. El valor *file* significa que s'utilitzarà un fitxer d'usuaris i opcionalment un de grups.

AuthUserFile: indica quin és el fitxer que conté els comptes dels usuaris locals del servidor Apache. És el fitxer que s'ha creat en l'exemple anterior.

AuthGroupFile: indica quin és el fitxer de grups en el qual consta quins grups d'usuaris hi ha i quins usuaris pertanyen a cada grup.

Require: aquesta directiva és la que determina quina és l'autorització que s'ha de fer. En l'exemple es permet que qualsevol usuari del grup profes tinga accés al recurs.

Finalment, cal verificar que l'accés al directori local és concedit únicament als membres del grup profes. Evidentment el mecanisme més senzill és verificar des d'un navegador l'accés al recurs www.dawcullera.org/daw i observar que es demana l'autenticació.

Exemples de mecanismes d'autorització

La directiva *require* és la que defineix l'autorització d'accés al recurs, és a dir, qui pot accedir-hi. Aquests en són alguns exemples d'ús:

1. *Require user valid-user*: permet l'accés a qualsevol usuari autenticat.
2. *Require user alumne1 alumne2*: permet l'accés als usuaris indicats (alumne1 i alumne2).
3. *Require group profes alumnes*: permet l'accés als usuaris que són membres d'algun dels grups indicats.

7. El protocol HTTPS

El protocol HTTP pateix els mateixos problemes de seguretat que els seus companys dels inicis d'internet (FTP, TFTP, SMTP…). Tota la informació viatja en text net i és fàcilment monitorable per altres. Quan un usuari es connecta a un web i indica l'usuari i la contrasenya, aquestes dades viatgen sense cap mena de protecció i qualsevol les pot capturar. Si el que es transmet són dades bancàries, llistes de contactes personals o qualsevol tipus de dada privada, és desaconsellable fer-ho per HTTP.

El primer mecanisme de seguretat que es va implementar per a HTTP va ser el protocol SSL (*Secure Socket Layer* o capa de sòcol segur), desenvolupat per Netscape. L'SSL proporciona una capa entre la capa de transport TCP i la capa d'aplicació HTTP en què les dades viatgen xifrades. L'HTTPS solament és un esquema URI que indica la utilització d'HTML més algun mecanisme de transport xifrat, com SSL o TLS.

Quan s'utilitza **HTTP amb un protocol xifrat** com SSL o TCL s'anomena HTTPS (*secure HTTP*). Utilitza el port 443.

El protocol SSL es va enviar a l'IETF (Internet Engineering Task Force o equip d'enginyeria d'internet, l'òrgan rector d'internet) per a l'estandardització i, després de diversos canvis, va sorgir el protocol TLS (*Transport Layer Security*, seguretat de capa de transport). El TLS proporciona les mateixes condicions de confidencialitat i autenticació en les transmissions HTTP que SSL.

Un dels avantatges de l'HTTPS és que permet la confidencialitat entre tots dos extrems de la comunicació, encara que només siga un dels extrems el que s'ha autenticat. Aquest model és molt pràctic quan, per exemple, un client anònim compra en un web autenticat.

Quan es volen pagar els bitllets d'avió, interessa que les dades de la targeta de crèdit viatgen xifrades i que el receptor siga la companyia aèria i no siga un web fals (un frau).

L'ús dels certificats no és exclusiu per autenticar el servidor. Si cal, els clients poden ser autenticats. Per exemple, un web pot requerir que els clients disposen del certificat que els atorga dret a accedir (expedit, per exemple, per la mateixa entitat).

Els passos necessaris per implementar comunicacions segures que permeten a un navegador client (o un client, siga qui siga) connectar-se via HTTPS a una seu web són:

- **Certificats digitals**: el servidor web ha de disposar d'una clau privada i d'un certificat digital.
- **Mòdul *mod_ssl***: cal tenir instal·lat el paquet de programari que proporciona les prestacions SSL al servidor i que la configuració activa carregue els mòduls pertinents.
- **Configuració de la seu web segura**: finalment, cal establir les directives SSL apropiades a la seu web que es vol configurar per fer-la accessible via SSL.

8. Certificats. Servidors de certificats

L'objectiu de les explicacions següents és implementar connexions segures HTTPS al web www.dawcullera.org utilitzant SSL com a mecanisme de transport xifrat.

Suposarem que el servidor disposa ja d'una clau privada i d'un certificat, amb independència de com s'haja obtingut. En concret, en el subdirectori certs del directori base del servei web hi ha:

- ***server.crt***: el fitxer corresponent al certificat o clau pública del servidor. Aquest fitxer assegura als clients que es connecten a la seu web que el servidor és qui realment diu ser.
- ***server.key***: és el fitxer amb la clau privada del servidor. Aquest fitxer s'ha codificat amb una *passfrase* o frase de pas de manera que cada vegada que s'inicialitzi el servidor web caldrà entrar aquesta frase.

Cal recordar que els navegadors clients validaran la confiança que els mereix el certificat contrastant el seu emissor amb la llista d'entitats certificadores que tenen carregada. Si l'emissor del certificat no hi és, caldrà fer passos per incorporar el certificat al navegador.

Aquests passos poden ser:

- Admetre el certificat com a vàlid quan el navegador presenta l'excepció de seguretat.
- Obtenir el certificat de l'entitat CA (*certification authority* o autoritat de certificació) que l'ha generat i incorporar l'entitat a la llista d'entitats en què el navegador confia.

Generar un certificat autosignat

A mode de recordatori ràpid, es pot generar una clau privada i un certificat autosignat fent:

openssl req new x509 nodes out server.crt keyout server. key

Configuració d'Apache per usar SSL

El servidor web podrà usar SSL si disposa dels mòduls que en proporcionen la capacitat. En cas de no tenir-los, cal buscar en els repositoris de programari habitual un paquet que proporcione el mòdul apropiat, instal·lar-lo i examinar el contingut. Usualment, tant el paquet com el mòdul que proporcionen les prestacions de trànsit segur SSL s'anomenen ***mod_ssl***.

Buscar el paquet mod_ssl i instal·lar lo.

```
root@a7b4cf9809a6:/# apt-cache search mod_ssl
```

libapache2-mod-gnutls - Apache module for SSL and TLS encryption with GnuTLS
libapache2-mod-nss - NSS-based SSL module for Apache2
python-mod-pywebsocket - WebSocket extension for Apache HTTP Server

```
root@a7b4cf9809a6:/# apt-get install libapache2-mod-gnutls
root@a7b4cf9809a6:/# a2enmod ssl
root@a7b4cf9809a6:/# service apache2 restart
```

Com podeu veure, el paquet conté, entre d'altres, un fitxer de configuració específic anomenat *ssl.conf* i un únic mòdul anomenat *mod_ssl*. Tant l'un com l'altre estan en el directori *mods-available*. El fitxer de configuració específic del mòdul SSL conté tot de directives que configuren el funcionament global del trànsit SSL. Com diu Apache, val més no tocar res. Es pot observar que el mòdul està carregat a la carpeta *mods-enabled:*

root@a7b4cf9809a6:/# ls -l /etc/apache2/mods-enabled/ssl.*
lrwxrwxrwx 1 root root 26 abr 15 10:38 /etc/apache2/mods-enabled/ssl.conf -> ../mods-available/ssl.conf
lrwxrwxrwx 1 root root 26 abr 15 10:38 /etc/apache2/mods-enabled/ssl.load -> ../mods-available/ssl.load

Configuració de la seu web amb SSL

Finalment cal aplicar a la seu web www.dawcullera.org les directives SSL apropiades per fer possible l'accés a aquesta seu web per HTTPS. El llistat de la directiva *VirtualHost* és:

```
<VirtualHost www.iocdaw.cat:443>
 ServerAdmin admin@ www.dawcullera.org
 DocumentRoot /var/www/daw
 ServerName www.dawcullera.org
 ErrorLog logs/m08 error_log
 CustomLog logs/daw access_log common SSLEngine On
 SSLProtocol all  SHAv2
 SSLCertificateKeyFile /var/www/certs/server.key
 SSLCertificateFile /var/www/certs/server.crt
 #SSLCACertificateFile /var/www/certs/ca.crt
</VirtualHost>
```

Les directives usades són:

- **Port 443**: aquest és el port usual per a les connexions segures HTTP. Si la seu web només escolta per aquest port només es podrà accedir al seu contingut per HTTPS. Si es volen seus diferents per al trànsit xifrat i per al no xifrat n'hi ha prou de crear una altra seu virtual amb un altre port.
- ***SSLEngine On***: aquesta directiva indica que cal activar el trànsit SSL per a aquesta seu web.
- ***SSLProtocolall-SHAv2***: en aquesta directiva s'indiquen quins protocols es poden usar per generar el trànsit xifrat. Les opcions *all* i *-SHAv2* indiquen que s'accepten tots els protocols vàlids excepte el protocol SHA versió 2.
- ***SSLCertificateKeyFile* <clau privada del servidor>**: aquesta directiva indica el fitxer amb la clau privada del servidor.
- ***SSLCertificateFile* <certificat>**: indica quin és el fitxer que conté el certificat del servidor. Aquest és el certificat que els navegadors clients veuran i del qual hauran de decidir si hi confien o no.
- ***SSLCACertificateFile* <certificat-CA>**: aquesta directiva és opcional i permet indicar quin és el fitxer que conté el certificat públic que ha emès l'entitat de certificació CA. Recapitulem, si el certificat del servidor ha estat emès per una entitat externa (per exemple, VeritatAbsoluta), aquesta directiva permet que el client obtingui el certificat de l'entitat emissora, estalviant-li la cerca. Ara bé, encara falta que el navegador client confie en aquesta entitat.

Un cop configurada apropiadament la seu virtual, cal posar de nou en funcionament el servei web (moment en què es demanarà la frase de pas del servidor per a la clau privada de www.dawcullera.org). Des de qualsevol navegador s'ha de poder accedir a la seu usant HTTPS. Ara bé, es generarà una excepció de seguretat perquè el navegador desconeix la procedència del certificat. Si l'usuari accepta confiar en la seu web, el certificat s'incorporarà al navegador i accedirà de forma xifrada a la seu web.

Una altra opció és carregar prèviament en el navegador el certificat de l'entitat CA VeritatAbsoluta, que és qui ha actuat en l'exemple com a entitat certificadora local. Si això es fa **abans** de contactar amb la seu web, quan el navegador hi accedeixi per HTTPS ja no es produirà una excepció de seguretat. Com que el certificat del servidor està signat per una entitat en la qual el navegador confia (forma part de la seva llista de *trusted Cas*) s'acceptarà automàticament.

Verificació de les connexions SSL

Els problemes principals que es poden trobar els navegadors en connectar amb seus web amb certificats són:

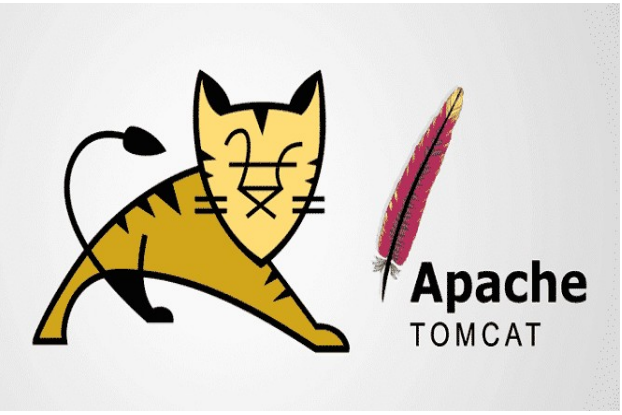
- Amb un certificat autosignat no cal definir cap CA. El navegador client mostrarà la típica pantalla d'excepció de seguretat i caldrà indicar que s'accepta el certificat de servidor per a l'entitat www.dawcullera.org. És un certificat emès per la mateixa entitat.
- Amb un certificat generat per una CA local cal incorporar manualment el certificat al navegador. Un cop fet això el navegador serà capaç de validar el certificat del

servidor amb la CA que l'ha expedit (*issuer*). En el nostre exemple, el certificat del servidor l'expedeix la CA VeritatAbsoluta.

A més dels navegadors, hi ha eines d'entorn de text per verificar connexions SSL, de la mateixa manera que s'utilitza *telnet host 80* per verificar connexions HTTP. D'una banda, es pot usar el mateix **OpenSSL** i, de l'altra, es pot instal·lar la utilitat **Curl**, que permet fer un ampli seguiment del diàleg SSL.

OpenSSL> s_client -connect www.iocdaw.cat:443 -state -debug
curl https://www.iocdaw.cat -kv

Tema 3: Administració de servidors d'aplicacions



Índex:

1. Introducció

El concepte de servidor d'aplicacions web és posterior al concepte de servidor web. En un primer moment, els servidors web només servien pàgines HTML estàtiques, sense oferir cap altre servei: les aplicacions web encara no existien. En aparèixer les primeres tecnologies de generació de contingut web dinàmic (CGI, PHP, ASP, JSP), apareix el concepte de servidor d'aplicacions web.

Alguns exemples que podem trobar són els següents:

- IIS: permet l'execució d'aplicacions web amb tecnologia ASPX (.NET Framework)
- Tomcat: integra el servidor web Apache amb un contenidor de *servlets* (JSP, Servlets)
- JBoss (Java Enterprise Edition)
- Oracle WebLogic: Java EE
- Websphere Application Server (IBM): Java EE
- LAMP (Linux Apache MySQL PHP)
- Zend Server (Apache+PHP)

Actualment és difícil poder distingir la frontera entre el servidor web i el servidor d'aplicacions. Les característiques i els serveis que ofereix el servidor d'aplicacions són els següents:

- Sistemes d'autenticació (seguretat)
- Implementació de lògica de negoci (per exemple, connexió amb motors de bases de dades)
- Gestió de sessions d'usuari
- Accés als components o llibreries de la plataforma utilitzada (Java o .NET)

Per als programadors (particularment quan treballen en web), el servidor d'aplicacions és un model molt interessant per alleugerir la feina de manteniment del sistema.

En aquest curs veurem diferents vessants dels servidors d'aplicacions. Treballarem tant la instal·lació i la configuració de servidors com la implantació dels fitxers. Alguns dels punts més importants que treballarem són:

Desplegaments de servidor LAMP dins d'un entorn UNIX

- Instal·lació d'un servidor LAMP
- Desplegament d'una aplicació en local (en el mateix servidor)
- Desplegament d'una aplicació en remot (amb diferents eines, com Putty i servidor FTP)

Aplicacions J2EE

- Instal·lació de Tomcat
- Administració de Tomcat
- Instal·lació d'una IDE (NetBeans) i integració amb Tomcat per a un treball més amigable
- Desplegament de *servlets* dins del servidor Tomcat
- Integració d'Apache i Tomcat fent servir els diferents mòduls disponibles

- Instal·lació d'un servidor Wildfly, conegut anteriorment com a JBoss

2. Servidors d'aplicacions web

El concepte de *servidor d'aplicacions web* és posterior al concepte de *servidor web*. En un primer moment, els servidors web només servien pàgines HTML estàtiques, sense oferir cap més servei: encara no existien les aplicacions web.

En aparèixer les primeres tecnologies de generació de contingut web dinàmic (CGI, PHP, ASP, JSP), apareix el concepte de **servidor d'aplicacions web**.

2.1. Desplegament en entorns LAMP

El terme LAMP fa referència a un grup de programari de codi lliure que s'instal·la normalment en conjunt per habilitar un servidor i allotjar llocs i aplicacions web dinàmiques. Aquest terme en realitat és un acrònim que representa un sistema operatiu Linux amb un servidor Apache, el lloc de dades és emmagatzemat en base de dades MySQL i el contingut dinàmic és processat amb PHP.

Alternatives a LAMP

Hi ha diverses alternatives per a LAMP:

- WAMP, només per a Windows
- MAMP, per a Windows i Mac
- XAMPP, per a Windows, Mac i Linux

Aquests programes faciliten les tasques per muntar el propi servidor web local només amb uns clics, tot i que alguns no són per a Linux.

2.1.1. LAMP

LAMP és l'acrònim usat per descriure un sistema d'infraestructura d'internet que usa les següents eines:

- Linux, el sistema operatiu en el que està recolzat.
- Apache, el servidor web.
- MySQL / MariaDB, el gestor de bases de dades.
- Perl, PHP, o Python, els llenguatges de programació.

La combinació d'aquestes tecnologies és usada principalment per a definir la infraestructura d'un servidor web, utilitzant un paradigma de programació per al desenvolupament.

Tots els **programes** esmentats anteriorment representen una **solució de treball** molt bona per a servidors web, sempre que treballin de forma conjunta.

2.1.2. Apache

El servidor HTTP Apache és un servidor web HTTP de codi obert, per a plataformes Unix (BSD, GN/Linux, etc.), Microsoft Windows, Macintosh i altres, que implementa el protocol HTTP/1.12 i la noció de lloc virtual.

El servidor Apache és desenvolupat i mantingut per una comunitat d'usuaris sota la supervisió de l'Apache Software Foundation, dins del projecte HTTP Server (httpd).

Apache presenta, entre altres característiques altament configurables, bases de dades d'autenticació i negociat de contingut, però va ser criticat per la falta d'una interfície gràfica que ajudés a configurar-lo.

2.1.3. MySQL

MySQL és un sistema de gestió de bases de dades relacional desenvolupat amb llicència dual GPL/Llicència comercial per Oracle Corporation. És considerada la base de dades de codi obert més popular del món i una de les més populars en general al costat d'Oracle i Microsoft SQL Server, sobretot per a entorns de desenvolupament web.

MySQL és usat per molts llocs web grans i populars, com Wikipedia, Google (tot i que no per a cerques), Facebook, Twitter, Flickr o YouTube.

2.1.4. PHP

PHP és un llenguatge de programació d'ús general de codi del costat del servidor originalment dissenyat per al desenvolupament web de contingut dinàmic. Va ser un dels primers llenguatges de programació del costat del servidor que es podien incorporar directament en el document HTML en lloc de cridar un arxiu extern que processés les dades. El codi és interpretat per un servidor web amb un mòdul de processador de PHP que genera la pàgina web resultant. PHP ha evolucionat i ara inclou també una interfície de línia de comandaments que es pot fer servir en aplicacions gràfiques independents.

2.1.5. Instal·lació d'un servidor LAMP

...

2.1.6. LAMP I phpMyAdmin

phpMyAdmin és una eina escrita en PHP amb la intenció de manejar l'administració de MySQL a través de pàgines web, utilitzant un navegador. Pot crear i eliminar bases de dades, crear, eliminar i alterar taules, esborrar, editar i afegir camps, executar qualsevol sentència SQL, administrar claus en camps; administrar privilegis i exportar dades en diversos formats.

Per poder utilitzar phpMyAdmin es requereix disposar d'un servidor web amb suport PHP i MySQL.

2.1.7. Desplegament d'una aplicació en l'àmbit LAMP

Es pot desplegar una aplicació sobre el servidor LAMP propi sobre:

- Desplegament de forma **local**
- Desplegament de forma **remota**

La principal diferència està en la forma de treballar sobre el servidor:

- En el desplegament local, es treballa directament sobre la màquina servidor; es pot fer un símil entre nosaltres i l'administrador local.
- En el desplegament de forma remota, cal connectar a la màquina de forma remota un gestor de transferència d'arxius, dotar-lo de seguretat i atorgar-li els permisos necessaris per a una correcta connexió.

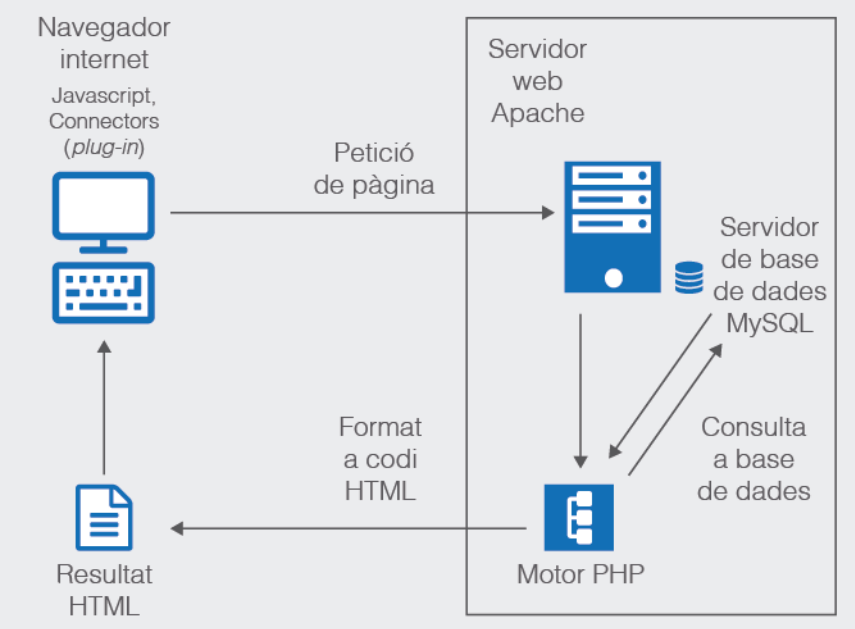
Alguns dels fitxers que haureu d'importar durant la tasca d'importació són els següents:

- Dades per a la base de dades
- Fitxers web

També heu de realitzar algunes accions bàsiques per a una correcta funcionalitat:

- Configurar la base de dades (podeu utilitzar un gestor).
- Fer el desplegament de fitxers web corresponents dins del servidor web. En cas d'una connexió remota, podeu utilitzar un servidor de transferència de fitxers.

Ací un esquema de l'arquitectura LAMP: arquitectura LAMP 'server'



2.2. Aplicacions J2EE

El terme servidor d'aplicacions s'aplica a totes les plataformes, tot i que està fortament identificat amb la plataforma J2EE. Tanmateix, també es refereix a servidors d'aplicacions basades en web.

Un **servidor d'aplicacions** és un motor que ofereix serveis d'aplicacions a clients o dispositius. El principal benefici d'un servidor d'aplicacions és la facilitat de desenvolupament de programari perquè les eines no necessiten ser programades: són assemblades a partir de la construcció de blocs sobre el servidor d'aplicacions.

Sobre la plataforma Java, fa referència al servidor d'aplicacions com un **servidor d'aplicacions J2EE**. Alguns exemples típics són WebLogic, Wildfly o Tomcat. Els mòduls

web són miniaplicacions de servidor (*servlets*) i JavaServer Pages, i la lògica de negoci és en Enterprise JavaBeans.

2.2.1. Servidor Tomcat

Tomcat és un contenidor web basat en el llenguatge Java que actua com a motor de *servlets* i JSP. S'utilitza per proveir *servlets* i Java Server Pages.

Tomcat proporciona un entorn per al codi Java per executar en cooperació amb un servidor web propi (per això també pot treballar com a servidor web independent). Tot i tenir el seu propi servidor, pot treballar combinat amb Apache Web Server.

Tomcat té un conjunt d'eines per configurar-lo, però també es pot configurar editant els fitxers de configuració que tenen el format XML.

Entorn Tomcat

El motor de *servlets* de Tomcat sovint es presenta en combinació amb el servidor HTTP Apache o altres servidors web. Tomcat pot funcionar com a servidor web per si mateix. En els seus inicis hi havia la percepció que l'ús de Tomcat de forma autònoma només era recomanable per a entorns de desenvolupament i entorns amb requisits mínims de velocitat i gestió de transaccions. Avui dia ja no hi ha aquesta percepció, i Tomcat és usat com a servidor web autònom en entorns amb alt nivell de trànsit i alta disponibilitat.

Tomcat està escrit en Java i funciona en qualsevol sistema que tingui una màquina virtual Java en funcionament; per tant, és multiplataforma.

Tomcat és un projecte de codi obert d'Apache Software Foundations que admet Java Servlet, Java Server Pages, Java EL, JSF i WebSocket.

Tomcat va ser seleccionat com a implementació de referència de contenidors de components web de Sun Microsystems (Servlet i JSP).

Funcionalitats i avantatges

Tomcat és un contenidor web i, per tant, està habilitat per realitzar les funcionalitats següents:

- Mapatge d'un URL a un *servlet*.
- Comprovació que la petició d'accés al *servlet* té els permisos d'accés correctes.
- Gestió de les peticions d'accés a *servlets* i JSP que, per exemple són responsables de carregar *servlets* en memòria, entre d'altres.
- Serveis de seguretat, concurrència, desplegament, gestió del cicle de vida, etc.
- Interfície entre el component web i la plataforma sobre la qual s'executa.
- Facilitat per desplegar un component web correctament empaquetat dins d'un fitxer .war.

Avantatges de fer servir Tomcat:

- És un servidor d'aplicacions de codi obert.
- És un *lightweight server* (no EJB).
- Té una integració fàcil amb Apache HTTP Server i amb IIS.

- És molt estable dins dels sistemes UNIX.
- Hi ha molts exemples i molta documentació en línia. És fàcil trobar repositoris disponibles.
- Implementa les especificacions de *servlet* i de JavaServer Pages (JSP) de Sun Microsystems.
- No requereix gaire memòria per a l'arrencada lleugera del sistema.
- És gratuït.

Estat del seu desenvolupament

Tomcat és desenvolupat i mantingut per membres de l'Apache Software Foundation i voluntaris independents. Els usuaris disposen de lliure accés al seu codi font i a la seva forma binària en els termes establerts en l'Apache License. Les primeres distribucions de Tomcat van ser les versions 3.0.x. Les versions més recents són les 8.x, que implementen les especificacions de Servlet 3.1 i de JSP 2.3. Les versions 4.0 i les posteriors utilitzen, internament, el contenidor de *servlets* Catalina.

Parts importants de Tomcat i estructura de directoris

Algunes de les parts més importants de Tomcat són les següents:

- **Catalina**: és el contenidor de *servlets*. Amb Catalina es poden implementar Realms per proporcionar sistemes d'autenticació i accedir a recursos per mitjà d'usuaris, contrasenyes i rols.
- **Coyote**: és un connector que admet HTTP1.1, que permet a Tomcat actuar com un servidor web.
- **Jasper**: compila fitxers JSP de manera que es puguin comportar com a *servlets* dirigits per Catalina. Necessita JDK 1.5 o superior.
- **Clusterització:** Tomcat permet la utilització de clústering, de manera que pot dirigir una gran quantitat de peticions de manera eficient.
- ***High availability***: permet actualitzar el servidor fàcilment sense afectar l'ús de les aplicacions desplegades.
- ***Load balancing***: permet distribuir la càrrega de feina entre diversos ordinadors, xarxes, CPU…

La jerarquia de directoris per defecte d'instal·lació de Tomcat inclou els següents:

- **/bin:** arrencada, aturada i altres scripts i executables.
- **/common:** classes comunes que poden utilitzar Catalina i les aplicacions web.
- **/conf:** fitxers XML i els corresponents DTD per a la configuració de Tomcat.
- **/logs:** logs de Catalina i de les aplicacions.
- **/server:** classes utilitzades solament per Catalina.
- **/shared:** classes compartides per totes les aplicacions web.
- **/webapps:** directori que conté les aplicacions web.
- **/work:** fitxers temporals, pàgines JSP precompilades i altres fitxers intermedis.

2.2.2. Instal·lar, configurar i administrar Tomcat

Tomcat es pot instal·lar en diferents sistemes operatius. No obstant això, sempre cal realitzar uns passos previs per al correcte desplegament i la implementació del servidor.

- Instal·leu Java (JDK): és fonamental i necessari perquè Tomcat pugui executar qualsevol aplicació (les aplicacions són codificades en Java).
- Descarregueu i establiu els paràmetres d'Apache Tomcat, el servidor s'escoltarà pel port 8080.
- Un cop instal·lat de forma correcta, agregueu Tomcat a un IDE (NetBeans, Eclipse…). D'aquesta manera podreu treballar d'una forma més còmoda i amigable.

Java Development Kit o (JDK) és un programari que proveeix eines de desenvolupament per crear programes en Java.

Entre els diversos rols que té Tomcat, podem trobar-hi els següents:

- Manager-gui: permet tenir accés a la interfície HTML.
- Manager-status: permet accedir únicament a la pàgina d'estat.
- Manager-script: permet accedir a les eines de text pla.
- Manager-JMX: permet accedir a la interfície JMX.

Entre totes les característiques que brinda, Apache Tomcat ve inclòs amb una aplicació web per poder configurar tant el servidor com les aplicacions del servidor. Per fer-ho cal seguir aquests passos:

- Desplegueu noves aplicacions web des de continguts carregats d'arxius .war.
- Llisteu els valors de les propietats dels sistemes operatius i les JVM.
- Llisteu els recursos JDNI globalment.
- Atureu una aplicació existent.

2.2.3. 'Servlets'

Els *servlets* van ser dissenyats per permetre l'extensió d'un servidor i proporcionar qualsevol servei. Actualment, però, només admeten HTTP i pàgines JSP. En el futur, un desenvolupador podria estendre un servidor FTP o un servidor SMTP utilitzant *servlets*. Un *servlet* amplia les funcionalitats d'un servidor oferint un servei específic dins d'un marc de treball ben definit. És una petita peça de codi Java (normalment una sola classe) que proporciona un servei específic.

Un *servlet* és una **llibreria per crear aplicacions web** en llenguatge Java. Sol utilitzar Tomcat per desplegar el projecte.

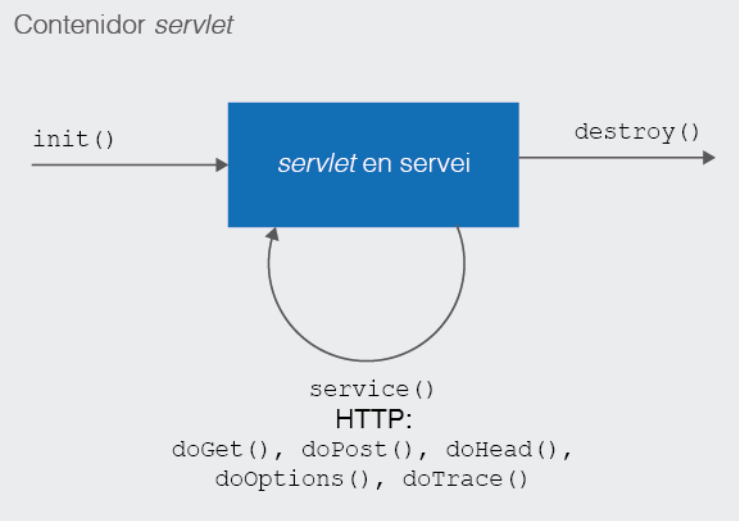
Per desplegar un *servlet* normalment es requereix la configuració d'un servidor d'aplicacions. Quan el servidor troba un tipus particular de sol·licitud invoca el *servlet*, i li passa els detalls sobre la sol·licitud i un objecte *response* per retornar el resultat.

Tots els *servlets* implementen la interfície javax.servlet.Servlet directament (en el cas dels genèrics) o indirectament (*servlets* HTTP o JSP). La interfície javax.servlet.Servlet inclou els següents mètodes importants:

- **init()**: defineix qualsevol codi d'inicialització que hauria d'executar quan es carrega el *servlet* en memòria.
- **service()**: és el mètode principal, cridat quan el *servlet* rep una sol·licitud de servei. Defineix un paquet de lògica de processament proporcionat pel *servlet*.
- **destroy()**: defineix qualsevol codi de neteja requerit abans d'eliminar la miniaplicació de la memòria.

Quan el contenidor *servlet* carrega per primera vegada un *servlet* invoca el mètode init() per inicialitzar. Després, a mesura que es fan sol·licituds per executar el *servlet*, el contenidor *servlet* invoca repetidament el mètode service(). Finalment, quan el contenidor *servlet* no necessita el *servlet*, crida el mètode destroy() i el descarrega de la memòria. Durant el temps de vida d'un simple exemplar *servlet*, els mètodes init() i destroy() només són invocats una vegada, mentre que el mètode service() és invocat moltes vegades (cada vegada que es faci una sol·licitud per executar la miniaplicació).

Figura del Cicle de vida del 'servlet'



2.3. Integració d'Apache i Tomcat

Actualment, la gran majoria de les organitzacions exposen la seva lògica de negoci a través de serveis web o aplicacions web. Per al correcte funcionament del negoci, és de vital importància que la gent pugui treballar sense errors informàtics o tecnològics.

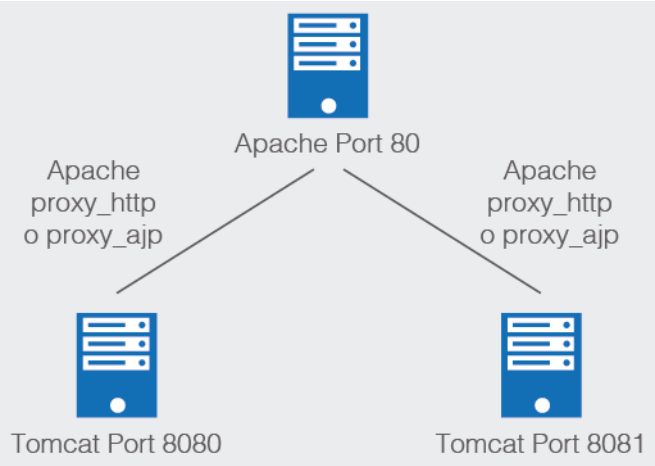
La realitat és que els sistemes fallen i cal evitar en la mesura del possible que aquests errors impedeixin accedir als serveis.

Configureu un conjunt de servidors perquè les peticions dels usuaris als serveis es distribueixin -a través d'alguna política- entre els servidors per aconseguir aquestes prestacions:

- **Alta disponibilitat**: en cas que un servidor caigui, un altre servidor actiu ha de prestar servei.
- **Balanç de càrrega**: cada servidor ha d'atendre un percentatge de les peticions, de manera que el sistema en conjunt admeti més usuaris.

2.3.1. Com treballar amb Apache i Tomcat

Hi ha dues opcions: redirigir el trànsit al connector HTTP de Tomcat o redirigir el trànsit al connector AJP de Tomcat. Conceptualment les dues opcions s'assemblen bastant i en entorns d'alt rendiment sembla que el connector AJP dona millors resultats. Podem observar en la figura els elements que intervenen per aconseguir l'objectiu: figura Arquitectura híbrida Apache i Tomcat.



A la imatge anterior podem observar els següents elements:

- Les aplicacions dels usuaris apunten a la direcció d'un balancejador (la resta és transparent, per a ells).
- El balanç (trànsit HTTP) està construït mitjançant el servidor web Apache i el mòdul mod_jk habilitat.
- El balanç, d'acord amb alguna política especificada a la configuració, distribueix el trànsit entre els usuaris (clients) i els servidors Tomcat.

Mòdul mod_proxy_http

Primer de tot, s'ha de tenir habilitat el mòdul Apache servidor intermediari HTTP (a2enmod proxy_http).

Per redirigir el trànsit, en la configuració d'Apache crea amfitrions (*hosts*) virtuals amb el format següent:

```
<VirtualHost *:80>
ServerAdmin admin@domini.com
ServerName domini.com
ServerAlias www.domini.com
ProxyRequests Off
Order deny,allow
Allow from all
ProxyPreserveHost on
ProxyPass / http://localhost:8080/
</VirtualHost>
```

Això farà que el trànsit es redirigeixi al port 8080 del propi servidor, de manera que dominio.com serà una instància Tomcat en la qual hi haurà configurat el seu connector HTTP per respondre a aquest port (en el fitxer server.xml).

<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

En aquesta configuració, i mentre el lloc tingui accessible el port 8080 des de l'exterior, l'aplicació serà accessible via domini.com:8080.

Mòdul mod_proxy_jk

El primer que cal fer és tenir habilitat el mòdul Apache intermediari AJP (a2enmod proxy_ajp).

La definició del sistema principal virtual en Apache és subtilment diferent:

```
<VirtualHost *:80>
ServerAdmin admin@domini.com
ServerName domini.com
ServerAlias www.domini.com
DocumentRoot /
ProxyPass / ajp://localhost:8009/
ProxyPassReverse / ajp://localhost:8009
</VirtualHost>
```

En Tomcat cal que estigui definit el connector AJP perquè accepti el trànsit al port 8009:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

En aquesta configuració es pot desactivar el connector HTTP de Tomcat de tal manera que ningú hi pugui accedir directament.

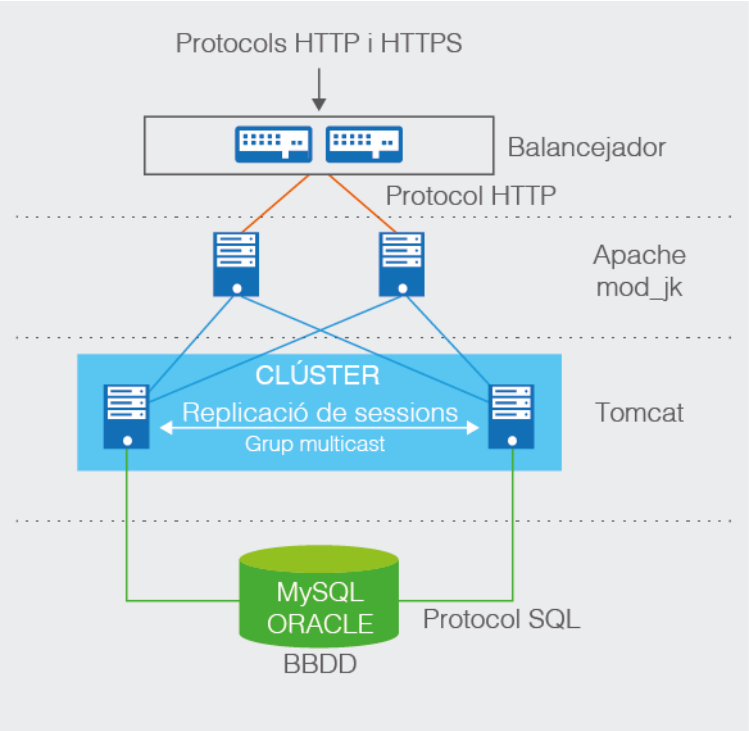
Qualsevol d'aquestes configuracions es podria replicar n vegades canviant el port amb el qual funciona cada Tomcat, i fins i tot es poden tenir aplicacions funcionant amb cada un dels dos models, basat en HTTP o basat en AJP.

L'**Apache JServ Protocol** (AJP) és, en el context de World Wide Web, un protocol binari que permet enviar sol·licituds des d'un servidor web a un servidor d'aplicacions que es troba darrere del servidor web.

Les funcionalitats bàsiques que podem trobar dins del model de desplegament mod_proxy_jk sobre Apache i Tomcat són:

- Les aplicacions dels usuaris apunten a la direcció d'un balancejador (la resta és transparent, per a ells).
- El balanç (trànsit HTTP) està construït mitjançant el servidor web Apache i el mòdul mod_jk habilitat.
- El balanç, d'acord amb alguna política especificada a la configuració, distribueix el trànsit entre els usuaris (clients) i els servidors Tomcat.

Vegeu en la figura l'esquema de desplegament del mòdul mod_proxy_jk sobre Apache i Tomcat



2.4. Servidor WildFly

WildFly, anteriorment conegut com a JBoss AS o simplement JBoss, és un servidor d'aplicacions Java EE de codi obert implementat en Java pur, més concretament, en l'especificació Java EE. Com que està basat en Java, JBoss pot ser utilitzat en qualsevol sistema operatiu per al qual estigui disponible la màquina virtual de Java.

L'URL www.jboss.org proveeix JBossDeveloper, el portal per a desenvolupadors de JBoss/WildFly, i wildfly.org passa a ser la web oficial del producte.

WildFly és programari lliure i de codi obert, subjecte als requisits de la GNU Lesser General Public License (LGPL), versió 2.1.

El 20 de novembre de 2014 JBoss Application Server es canvia el nom a WildFly. La JBoss Community i altres productes JBoss de Red Hat com JBoss Enterprise Application Platform no es canvien el nom. Malgrat el canvi, JBoss segueix essent el 2016 el terme més usat per anomenar el producte, tant en termes de treball com de web.

2.4.1. Funcionalitats

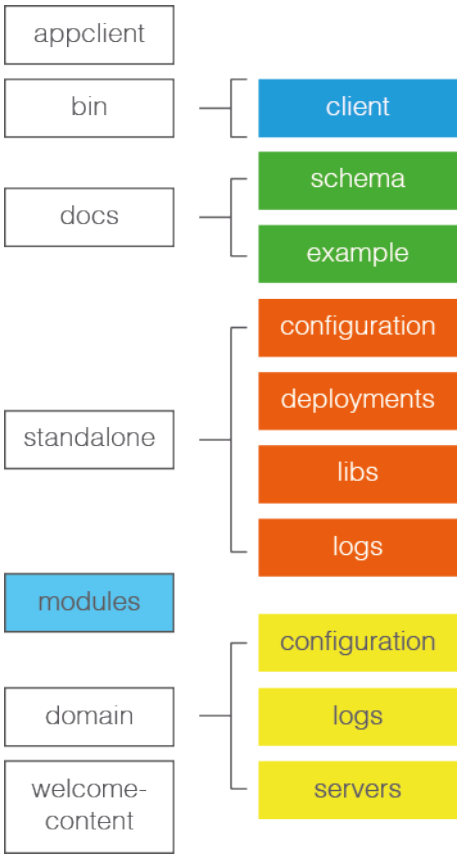
JBoss AS és el primer servidor d'aplicacions de codi obert, preparat per a la producció i certificat J2EE 1.4, disponible al mercat, i ofereix una plataforma d'alt rendiment per a aplicacions d'*e-business*. Combina una arquitectura orientada a serveis SOA, amb una llicència GNU de codi obert, i té les funcionalitats següents:

- Clusterització
- Recuperació davant fallades

- Balanç de càrrega
- *Caching* distribuït
- Implantació (programari) distribuïda
- Enterprise JavaBean

2.4.2. Estructura

Baix de la carpeta principal creeu els següents directoris. Figura de disposició dels fitxers del servidor Wildfly



Aquesta descripció és igual per a Linux que per a Windows.

- **appclient**: conté fitxers de configuració, contingut d'implementació i àrees d'escriptura utilitzades pel contenidor del client d'aplicació que s'executa des d'aquesta instal·lació.
- **bin**: conté scripts, configuració inicial d'arxius i diversos comandaments de línia útils (vault.sh, add-user.sh).
- **docs/schema**: conté l'*schema* XML de definicions.
- **docs/examples/config**: conté algunes configuracions en solitari (tal com standalone-minimalistic.xml).
- **domain**: conté els arxius de configuració del domini, external image arrow-10×10.png del servidor i àrea d'escriptura usada pel domini.
- **modules**: conté tots els mòduls.
- **standalone**: conté l'arxiu de configuració, contingut del desplegament i àrees escrivibles usades pel servidor en solitari corrent per a l'aplicació.

- **welcome-content**: conté contingut relacionat amb l'aplicació web predeterminada (*root*).

Les carpetes bin i servers seran les més utilitzades, perquè s'hi allotjaran els arxius executables i les aplicacions, respectivament.

Si no hi ha errors, en executar run.bat (Windows) o RUN.SH (Linux) accedireu al **Jboss**, posant en un navegador l'adreça IP on està instal·lat el servidor amb el port 8080.

2.4.3. Instal·lar WildFly

A forma d'introducció, cal veure com fer una instal·lació des de zero. A més, aprendreu com iniciar i aturar el servidor i veureu algunes funcionalitats bàsiques del servidor.

Es necessita:

- JDK instal·lat a l'equip
- Descàrrega del servidor d'aplicacions

Arrencada del servidor

Per iniciar el servidor situeu-vos en el directori */bin* de la instal·lació de WildFly i executeu-lo. Proveu el seu desplegament obrint un navegador i sol·licitant l'URL http:/localhost:8080.

Aturada del servidor

Per aturar el servidor executeu la seqüència **Ctrl + C** a la finestra on s'està executant l'script d'execució (startup.bat), o bé l'script per aturar (shutdown.bat) situat en el mateix directori l'script d'arrencada (/bin).

Tema 4. Instal·lació i administració de servidors d'FTP

Índex

1. Introducció

Les aplicacions de transferència de fitxers van ser una de les primeres eines en desenvolupar-se en l'expansió de les xarxes d'internet. La necessitat de poder accedir a diferents sistemes i intercanviar informació va originar un dels sistemes que actualment es fan servir.

Actualment hi ha diferents formes d'intercanvi d'informació de forma distribuïda en format fitxer:

- Sistemes de fitxers en xarxes
- Programari de missatgeria
- Programari de distribucions de fitxers P2P (*peer-to-peer*)

P2P. El P2P és un concepte de xarxes de computadors referent a la transmissió entre dos equips dins la xarxa. Concepte aplicat en les aplicacions com Napster, eMule, Bittorrent, en la transmissió de fitxers en les xarxes d'internet entre clients.

Windows 10 fa servir tecnologia P2P per realitzar la distribució de les actualitzacions del sistema operatiu dins d'una xarxa local.

L'FTP (*file transfer protocol*) o **protocol de transferència de fitxers** és un protocol que proporciona el servei de transferència de fitxers entre sistemes de diferent naturalesa, és a dir, es poden interconectar clients de Linux cap a un sistema de Microsoft o d'altres.

La implementació de l'FTP es remunta a l'any 1971, quan es va desenvolupar un sistema de transferència de fitxers, definit dins la RFC **(***request for comments***) 141**, entre equips de l'Institut Tecnològic de Massachusets (MIT, Massachusetts Institute of Technology). Durant els anys posteriors es van fer diferents innovacions al protocol bàsic, que es van incloure l'any 1973.

El protocol FTP, tal com es coneix actualment com a estàndard, s'especifica dins la RFC 959 l'any 1985 i defineix el funcionament del protocol. Posteriorment, el protocol FTP s'ha anat revisant amb algunes noves característiques, però la seua base de funcionament s'ha mantés.

RFC. Els *request for comments* o **documents** RFC són una sèrie de publicacions que descriuen diversos aspectes del funcionament d'internet, xarxes de computadors, protocols i procediments. La seva creació, per part de Steve Crocker l'any 1969, estava destinada al registre dels dissenys del grup de treball de xarxa per a ARPANET. El registre s'efectua basat en un esquema per realitzar el contingut i usant text en format ASCII (American Standard Code for Information Interchange).

Protocol d'internet. Conjunt de regles de comunicació de xarxa en què es basa internet i que, a més, faciliten l'intercanvi de dades entre ordinadors connectats a la xarxa.

El protocol FTP es basa en l'arquitectura client/servidor i fa ús del protocol de control de transport, TCP (*transport control protocol*) per realitzar el canal de transmissió entre el client i el servidor, amb la garantia que la informació que s'envia o es llegeix arribarà al seu destí.

TCP. El TCP (*transmission control protocol*) és un protocol de control de transmissió dissenyat l'any 1973 i 1974 per Vint Cerf i Robert Kahn que es defineix dins la capa de transport en la pila de protocols TCP/IP. Les especificacions es troben dins la RFC 793 i la RFC 1323.

Es fan servir dos canals de comunicació dins del protocol FTP, el canal de control i el canal de dades:

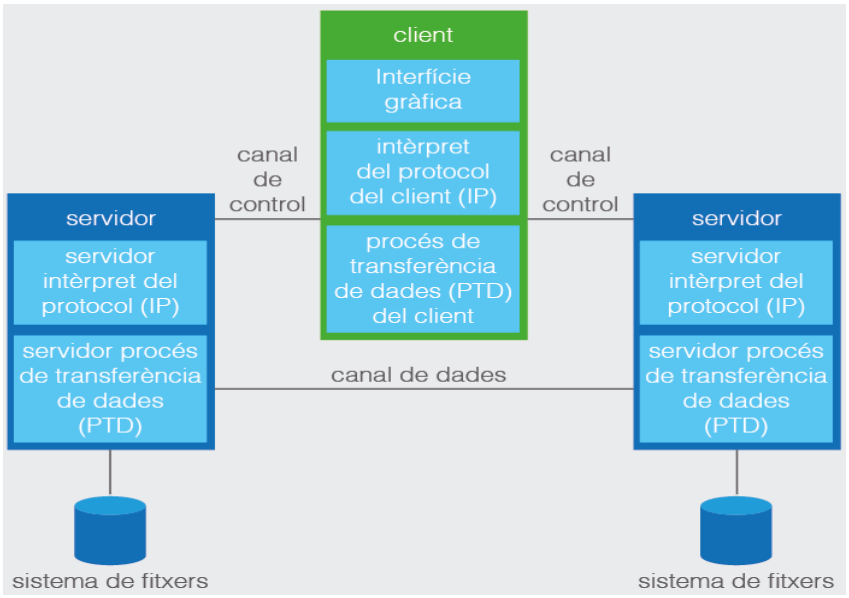
- El **canal de control** envia totes les ordres de comunicació, com poden ser iniciar la sessió de treball i ordres d'execució com llegir, escriure, llistar, esborrar, etc.
- El **canal de dades** envia el contingut d'aquells fitxers a treballar, que pot ser tant per llegir el contingut del fitxer com per fer l'escriptura del fitxer.

Tant el client com el servidor gestionen dos processos:

No confongueu les sigles IP (intèrpret de protocol) amb l'Internet Protocol.

- **PTD** (procés de transferència de dades): és l'encarregat d'establir la connexió i administrar el canal de dades. Tant el client com el servidor tenen el seu propi PTD.
- **IP** (intèrpret del protocol): interpreta el protocol i permet que el PTD pugui ser controlat mitjançant ordres rebudes pel canal de control.

L'IP és diferent en el client i servidor, cadascú s'encarrega d'unes funciones específiques. A la següent figura podeu observar l'estructura de funcionament del client i servidor FTP juntament amb el procés IP i PTD



L'IP del servidor:

- Escolta les ordres que provenen de l'IP de l'usuari mitjançant el canal de control per un port de dades.
- Estableix la connexió del canal de control.
- Rep les ordres FTP de l'IP de l'usuari, les respon i executa al PTD del servidor.

L'IP del client:

- És el responsable d'establir la connexió amb el servidor FTP.
- Envia ordres FTP.
- Rep les respostes del servidor IP.
- Controla el PTD de l'usuari.

Quan un client connecta al servidor FTP, l'IP de l'usuari inicia la connexió amb el servidor amb el protocol Telnet (RFC 854).

Telnet. Telnet és un protocol que permet connectar-se a sistemes remots anomenats *hosts* fent servir la xarxa TCP/IP (xarxes LAN o internet). Mitjançant un programa anomenat client de Telnet, es pot fer una connexió a un servidor Telnet remot.

Una vegada iniciada la comunicació amb el servidor Telnet, s'obté un terminal virtual que permet la comunicació amb el servidor Telnet o *host* remot.

Per poder accedir a un *host* remot es necessita un usuari i una contrasenya que ha de ser dins del sistema remot.

L'especificació del protocol Telnet es troba dins la RFC 854.

El client envia ordres FTP al servidor, el servidor les interpreta, executa el PTD i respon amb un format estàndard. Una vegada establerta la connexió, l'IP del servidor proporciona el port pel qual s'enviaran les dades al PTD del client, per on escoltarà i rebrà les dades del servidor. El sistema d'emmagatzematge dels fitxers dependrà del sistema on siga i el seu sistema de fitxers. Per exemple, sistemes de fitxers ext4 per a Unix o NTFS per a Microsoft Windows.

NVT-ASCII. Sistema estàndard definit dins del sistema Telnet per a l'enviament d'ordres i dades fent servir el sistema de codificació de caràcters ASCII.

Tota la comunicació que es fa en el canal de control segueix les recomanacions del protocol Telnet. Les ordres FTP són cadenes de caràcters Telnet amb format NVT-ASCII (*Network Virtual Terminal-American Standard Code for Information Interchange*) per on s'envien les ordres de l'FTP.

Les ordres FTP permeten especificar:

- El port que es farà servir.
- El mètode de transferència de dades.
- L'estructura de dades.
- L'acció que es durà a terme (llegir, eliminar, llistar, emmagatzemar, etc.).

Hi ha tres distribucions d'ordres FTP:

- **Ordres de control d'accés**: especifiquen els identificadors del control d'accés. La majoria d'aquests controlen qui accedeix al servidor FTP i quins privilegis tindrà l'usuari.
- **Ordres de paràmetres de transferència**: tenen un valor per defecte del servidor FTP, i només es fa ús d'aquests paràmetres si han estat modificades.
- **Ordres de servei** FTP: són les ordres més usades. Defineixen la transferència de fitxers i la navegació dels directoris remots per a l'usuari. L'argument principal de

les ordres de servei sol ser el nom d'un directori. Totes les dades que s'envien per a una ordre de servei sempre s'envien pel canal de dades.

2. Configuració del servei de transferència de fitxers. Permisos i quotes

Actualment hi ha moltes aplicacions que implementen el protocol FTP tant per la banda del client com per la del servidor. D'aquestes implementacions del protocol FTP n'hi ha de font pública i que es poden baixar gratuïtament en sistemes propietaris o lliures. La decisió d'una aplicació o una altra que implemente el protocol FTP ve donada per les possibilitats que ofereix i el sistema de treball on s'exerceix la feina. En el cas del desplegament d'aplicacions web, qualsevol servidor o client FTP s'ajusta a les necessitats del desplegament web.

Per defecte la majoria de sistemes operatius porten un client FTP gràfic o terminal, per poder accedir remotament als servidors externs.

2.1. Configuració del servei de transferència de fitxers

En aquesta unitat fareu servir el servidor FTP anomenat **ProFTPD** (*short for PRO* FTP *Daemon*). ProFTPD és un servidor FTP amb llicència GPL (*general public license*) per a Linux que permet fer una customització del seu funcionament depenent de les necessitats de configuració de l'entorn de treball per part de l'administrador.

Les característiques principals d'aquest programari són:

- El seu sistema de configuració es basa en un fitxer de configuració amb directrius intuïtives molt semblants a les del servidor Apache Web Server ja vist.
- Permet la configuració de servidors virtuals.
- Permet l'execució del servei com a servidor independent (*standalone*) o *inetd*.
- Permet mantenir l'arrel del directori anònim.
- El codi font està disponible per als desenvolupadors.
- Permet amagar els fitxers i directoris, basant-se en els permisos que fa servir Linux.

Dimonis 'standalone' i 'inetd'

El dimoni *standalone* escolta les peticions del servei i llança diferents processos per tractar-les. Es fa servir per a trànsits elevats de dades i usuaris. El servei sempre està actiu.

El dimoni *inetd* escolta les peticions del servei dels ports. Si detecta comunicació en el port configurat, inicia el servei passant-li la connexió. Es fa servir en situacions de poc trànsit.

Hi ha una gran varietat de ProFTPD en distribucions basades en Unix excepte plataformes Microsoft, encara que és possible que es puga3. executar gràcies a l'acord Ubuntu Microsoft, que permet executar el Terminal Bash al sistema Windows 10 des de l'any 2016, amb el programa desenvolupador.

3. Configuració de ProFTPD

El sistema on es fa la instal·lació i configuració està basat en un sistema Debian (Ubuntu o Lubuntu). El conjunt d'instruccions d'instal·lació o configuració mitjançant ordres de

sistema en són dependents (si anem a un sistema REDHAT, CENTOS, etc., haureu de mirar el procediment d'instal·lació i les ordres específiques del sistema).

La configuració interna és independent del sistema operatiu usat, qualsevol configuració que feu durant aquests apartats és compatible amb altres sistemes operatius basats en Unix.

Per iniciar el procés d'instal·lació de ProFTPD, primer de tot obriu un terminal i executeu l'ordre d'actualització dels repositoris d'Ubuntu:

```
sudo apt-get update
sudo apt-get install proftpd
```

Una vegada accepteu la instal·lació, el procés APT instal·la proFTPD i us fa escollir entre dues opcions referents a la modalitat d'inici del servei.

Per defecte, escolliu proFTPD com a procés indepenent (*standalone*).

```
root@7d6238b31227:/# service proftpd status
ProFTPD is started in standalone mode, currently running.
root@7d6238b31227:/#
```

El fitxer */etc/shadow* conté els noms dels usuaris del sistema on treballeu.

Una vegada finalitzada la instal·lació, procediu a verificar el resultat d'executar unes ordres. Executeu l'ordre següent:

```
sudo cat /etc/shadow
```

L'ordre que acabeu d'executar llista tots els usuaris del sistema on treballeu. Si us hi fixeu bé trobareu una línia anomenada ftp i proftpd. Aquestes línies fan referència a dos usuaris que ha creat el procés de configuració inicial de proFTPD. Els usuaris ftp i proftpd són usuaris d'accés per realitzar accessos anònims.

Podeu comprovar el mateix resultat amb les ordres següents:

sudo cat /etc/shadow | grep proftpd
sudo cat /etc/shadow | grep ftp

O amb una ordre única:

```
sudo cat /etc/shadow | grep -E "proftpd|ftp"
```

```
root@7d6238b31227:/# cat /etc/shadow | grep -E "proftpd|ftp"
proftpd:!:19666:0:99999:7:::
ftp:*:19666:0:99999:7:::
root@7d6238b31227:/#
```

A continuació comproveu l'estat dels processos realitzats amb ProFTPD:

```
sudo ps -ef | grep proftpd
```

Tindreu la següent eixida per terminal:

```
root@7d6238b31227:/# ps -ef | grep proftpd
proftpd      457       1  0 20:29 ?        00:00:00 proftpd: (accepting connections)
root         478       1  0 20:34 pts/0    00:00:00 grep --color=auto proftpd
root@7d6238b31227:/#
```

Veureu que hi ha un procés anomenat proftpd i que està acceptant les connexions. A continuació, executeu l'ordre:

```
sudo netstat  -ltn
```

Aquesta ordre us mostrarà aquells processos que estan escoltant el port 21 (port del servei FTP).

La línia que us interesa reconeixer és l'última, ja que confirma que hi ha un procés que està escoltant el port 21, que és el que es fa servir dins del protocol FTP.

```
root@7d6238b31227:/# netstat  -ltn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp6       0      0 :::21                   :::*                    LISTEN
root@7d6238b31227:/#
```

Una vegada heu verificat els processos i els ports de proFTPD, la configuració del servei FTP es realitzarà mitjançant un fitxer de text pla, tal com se sol fer amb molts serveis d'Unix, com per exemple Apache Web Server.

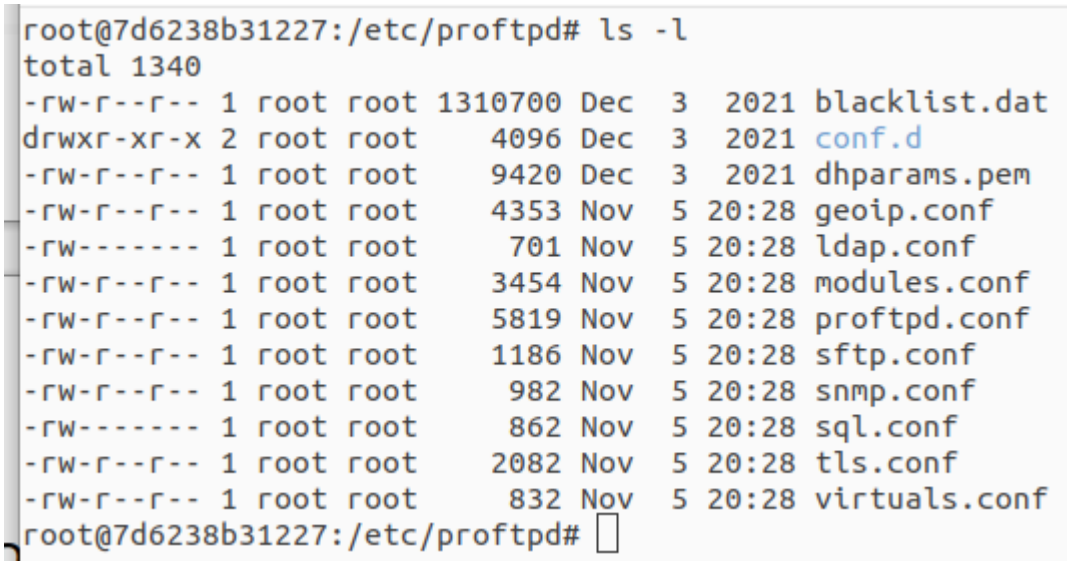
El directori on es troben les configuracions són dins de */etc/proftpd/* i el fitxer de configuració principal és el fitxer *proftpd.conf*.

Editar el fitxer original pot comportar haver de modificar el fitxer i perdre les dades inicials de configuració. Sempre és bo fer una **còpia de seguretat** del fitxer. Amb l'ordre cp (per copiar fitxers) o navegant al directori */usr/share/proftpd/templates/* teniu les còpies dels fitxers originals, en cas de voler recuperar les configuracions inicials.

Navegueu fins al directori de configuracions de ProFTPD */etc/proftp* i executeu:

```
ls -l
```

Amb aquesta ordre llistareu tot el contingut del directori, amb el resultat següent:



Com podeu veure, el contingut del directori */etc/proftpd* conté diferents fitxers, inclòs el proftpd.conf. Els fitxers que hi ha dins fan referència a fitxers de configuracions de modularitats que amplien la capacitat de ProFTPD, com poden ser vinculacions de dades amb SQL, configuracions de seguretat per xifrar la informació, creació de servidors virtuals, vinculació amb el servei de directori LDAP (*Lightweight Directory Access Protocol*), etc.

Inicieu l'edició del fitxer *proftd.conf*. Dins del directori feu:

```
sudo nano proftpd.conf
```

En les següents ordres dins d'aquest apartat es fa servir l'editor de text Nano, però podeu editar els fitxers de configuracions amb diferents editors dins del terminal o gràfics.

Abans de detallar configuracions, fixeu-vos en el fitxer *proftpd.conf*. Moltes de les línies del fitxer de configuració estan iniciades amb el caràcter # (*hashtag* o etiqueta). El caràcter # es fa servir en múltiples configuracions de serveis del sistema i permet comentar línies de configuració i fer comentaris. L'intèrpret de l'aplicació no els executarà.

Com podeu comprovar, el fitxer de configuració *proftpd.conf* és bastant extens. Tot seguit es detallen les opcions de configuració més rellevants.

ProFTPD té una gran extensió de directives i per la seva extensió no es pot detallar tot.

Dins de la configuració inicial, les directives més importants són:

- *Servername*: defineix el nom del servidor que mostrarà, en aquest cas "Debian". Podeu cambiar el nom per un que identifiqui el nostre servei.
- *TimeoutIdle*: defineix el temps que un usuari pot estar connectat sense fer cap acció.
- *TimeoutStalled*: defineix el temps que una connexió de dades pot estar aturada.
- *DisplayLogin*: defineix el fitxer de text on es mostrarà el missatge de benvinguda.
- *DisplayChdir*: defineix el fitxer de missatge que es mostrarà a cada canvi de navegació de directori.
- *ListOptions*: defineix l'ordre "-l" per a llistar els directoris.

- *DefaultRoot*: encapsula els usuaris en els directoris *home* de cada usuari.
- *RequireValidShell*: si el valor és *on* obliga que els usuaris de sistema tinguen definit un *shell* vàlid a la seva configuració.
- *Port*: defineix el número de port que es farà servir per a les connexions de control, per defecte el 21.
- *Umask*: màscara de permisos que tindrà per defecte.
- *AllowOverWrite*: permet sobreescriure el fitxer si el valor està a \on\.
- *QuotaEngine*: permet activar el motor de quotes del servidor FTP.

Aquestes directives estan actives sense el símbol *tag* i funcionen per defecte en el servidor.

Les següents directives amb el símbol *tag*:

Per ampliar la teoria, vegeu l'apartat "Protocol de transferència de fitxers segur".

- *#Include /etc/proftpd/virtuals.conf*: fitxer on s'habilitaran les configuracions de servidors virtuals alternatius que es poden configurar.
- *#Anonymous*: habilita el servidor anònim.
- *#Include /etc/proftpd/tls.conf*: fitxer on es configurarà el servidor per suportar connexions segures.
- *#Directory*: configura un directori específic amb una configuració específica, com pot ser habilitar accés als usuaris, permetre llegir, escriure, llistar, etc.

4. Permisos

Dins d'un servei FTP, un dels passos importants és el de concedir permisos determinats per controlar l'accés al servidor o als diferents directoris.

Dins de ProFTPD la directiva **<LIMIT>** permet fer les configuracions de permisos dins del servidor FTP.

La directiva **<LIMIT>** la podem configurar dins de les directives. Dins de la configuració general del servidor:

- *<VirtualHost>*
- *<Directory>*
- *<Anonymous>*
- *<Global>*

I els permisos generals que podem configurar són:

- *ALL* (tots els permisos excepte de LOGIN).
- *DIRS* (inclou **CDUP, CWD, LIST, MDTM, MLSD, MLST, NLST, PWD, RNFR, STAT, XCUP, XCWD, XPWD**)
- *LOGIN* (accés d'usuaris)
- *READ* (inclou **RETR, SIZE**)
- *WRITE* (inclou **APPE, DELE, MKD, RMD, RNTO, STOR, STOU, XMKD, XRMD**)

Vigileu quan gestioneu permisos de directoris de sistema amb CHMOD. Reviseu pertinences de grups i permisos dels usuaris. Una màscara 770 ens donarà permís total

(escriptura, lectura i execució) a propietaris i al grup, la 774 permet la lectura a la resta d'usuaris i la 777 donaria permisos a tothom, eviteu fer servir aquest últim. El principal receptor de la gestió d'aquests permisos és la directiva *<Directory>*.

Aquestes paraules clau (ALL, DIRS, LOGIN, READ, WRITE) permeten configuracions generals, però podem també configurar permisos específics que estan englobats dins les generals.

Altres directives útils que podeu fer servir dins de <LIMIT> són:

- *AllowUser nomUsuari*: dona el permís a un usuari específic.
- *DenyUser nomUsuari*: denega el permís a un usuarui específic.
- *AllowAll*: dona el permís a tots els usuaris.
- *DenyAll*: denega el permís a tots els usuaris.

Vegeu alguns exemples de configuracions, que poden anar dins del fitxer *proftpd.conf*:

```
#permet entrar al servidor FTP a l'usuari 1 i 2 i denegar l'accés a la resta
<LIMIT LOGIN>
  AllowUser usuari1
  AllowUser usuari2
  DenyAll
</LIMIT>

#modifiquem els permisos del directori /var/ftp/dades
<Directory /var/ftp/dades>
  #es dona permisos de lectura a l'usuari 1 i 2 i es nega la lectura de fitxers
a la resta
  <LIMIT READ>
    AllowUser usuari1
    AllowUser usuari2
    DenyAll
  </LIMIT>
  #es dona permisos d'escriptura a l'usuari 1, es nega l'escriptura de fitxers a
l'usuari 2 i a la resta d'usuaris
  <LIMIT WRITE>
    AllowUser usuari1
    DenyAll
  </LIMIT>
</Directory>
```

Sempre que vulgueu configurar els permisos d'un directori de ProFTPD necessitareu crear la directiva *<Directory>* i indicar la configuració dels permisos que vulgueu editar.

5. Quotes

ProFTPD permet configurar quotes d'espai diferenciant entre quotes generals del servidor, quotes vinculades a usuaris o grups de treball.

Les quotes generals del servidor permeten configurar:

- Restringir la velocitat de pujada i de descàrrega dins del servidor FTP.
- Restringir el màxim d'espai d'emmagatzematge d'un fitxer al servidor FTP.
- Restringir el màxim de la mida del fitxer que podem descarregar del servidor.

Les quotes d'usuari o grups de treball permeten:

- Restringir la velocitat de pujada i de descàrrega de l'usuari o el grup de treball.
- Restringir el màxim d'espai d'emmagatzematge d'un fitxer per part de l'usuari o del grup de treball.
- Restringir el màxim de la mida del fitxer que pot descarregar l'usuari o el grup de treball.
- Restringir d'espai propi per emmagatzemar dades en el directori de configuració de l'usuari o del grup de treball.

5.1. Quotes generals del servidor ProFTPD

Per realitzar la configuració i limitar les quotes generals a tots els usuaris, cal afegir les directives.

Per configurar el màxim de fitxer d'emmagatzematge dins del servidor FTP:

#Unitats amb case-insensitive "Gb" (Gigabytes), "Mb" (Megabytes), "Kb" (Kilobytes), o "B" (bytes)
MaxStoreFileSize 20 Mb

Per configurar el màxim de descàrrega d'un fitxer del servidor FTP:

#Unitats amb case-insensitive "Gb" (Gigabytes), "Mb" (Megabytes), "Kb" (Kilobytes), o "B" (bytes)
MaxRetrieveFileSize 20 Mb

5.2. Quotes d'usuari o grups de treball a ProFTPD

ProFTPD fa servir dos tipus de configuracions de quotes dins de dos fitxers:

- *limit*: per fixar els màxims de descàrrega, pujada, ràtios, etc.
- *tally*: per portar el compte de la quantitat fins al moment.

Per realitzar les quotes d'usuari, cal fer servir les ordres de terminal que implementa ProFTPD per configurar les quotes.

Per crear el fitxer *limit* i *tally*, feu el següent dins del terminal Linux i dins del directori de treball on s'emmagatzemaran els fitxers:

```
cd /etc/proftpd
mkdir /etc/proftpd/quotes
cd /etc/proftpd/quotes
ftpquota --create-table --type=limit
ftpquota --create-table --type=tally
```

Aquestes ordres donen com a resultat els fitxers següents:

1. *ftpquota.limittab*
2. *ftpquota.tallytab*

```
root@7d6238b31227:/etc/proftpd/quotes# ftpquota --create-table --type=limit
ftpquota --create-table --type=tally
root@7d6238b31227:/etc/proftpd/quotes# ls -l
total 8
-rw-r--r-- 1 root root 4 Nov  5 20:59 ftpquota.limittab
-rw-r--r-- 1 root root 4 Nov  5 20:59 ftpquota.tallytab
root@7d6238b31227:/etc/proftpd/quotes#
```

Dins del fitxer de *proftpd.conf* afegiu les següents línies dins del servidor principal o, si esteu configurant un servidor virtual:

```
<IfModule mod_quotatab_file.c>
  QuotaEngine on
  #mostrar a l'usuari les unitats de la quota "b"|"Kb"|"Mb"|"Gb"
  QuotaDisplayUnits Mb
  # Permet fer l'ordre quote SITE QUOTA
  QuotaShowQuotas on
  QuotaLog /var/log/proftpd/quota.log
  QuotaLimitTable file:/etc/proftpd/quotes/ftpquota.limittab
  QuotaTallyTable file:/etc/proftpd/quotes/ftpquota.tallytab
</IfModule>
```

La instrucció *<IfModule mod_quotab_file.c>* permet que s'execute el conjunt de configuracions que hi ha dins d'aquesta etiqueta, sempre que el *mod_quotab_file.c* estiga habilitat.

Detall de les directives:

- *QuotaEngine*: permet iniciar el sistema de quotes amb el valor *on*.
- *QuotaDisplayUnits unitat*: permet mostrar l'estat de la quota en la unitat d'emmagatzematge que especifiquem, *"b"|"Kb"|"Mb"|"Gb"*.
- *QuotaShowQuotas*: habilitem l'ordre *SITE QUOTA* dins d'FTP. Permet mostrar l'estat de la quota i habilitar amb el valor *on*.
- *QuotaLog*: permet configurar el directori de registres de Log.
- *QuotaLimitTable i QuotaTallyTable*: especifica on es troben els fitxers de quota creats amb les ordres *ftpquota*.

Una vegada creats els fitxers de quotes i afegida la configuració bàsica al fitxer *proftpd.conf*, executeu les ordres per crear la quota a un usuari específic.

Executeu l'ordre següent a tants usuaris o grups com s'hagen de configurar:

```
ftpquota --add-record --type=limit --name=nomUsuari --quota-type=user --units=B
--bytes-download=15728640
```

Per a més detall de l'ordre *ftpquota* i les opcions de quotes adreceu-vos a les configuracions de quotes de ProFTPD

El detall de l'ordre:

- *–add-record*: afegeix un registre al fitxer de quotes que s'espicificarà.
- *–type*: especifica a quin fitxer anirà el registre, fitxer *limit* o *tally*.
- *–name*: especifica el nom de l'usuari o grup a qui és aplicable la quota.
- *–quota-type*: especifica si la quota és per usuari o grup, *user* o *group*.

- *–units*: unitats que es faran servir per calcular les unitats d'emmagatzematge i processament de dades, "B" o "byte", "Kb" o "kilo", "Mb" o "mega", i "Gb" o "giga". Per defecte, fa servir "byte" si no es fa servir la directiva *units*.
- *–bytes-download*: especifica el nombre màxim de bytes que s'han de descarregar de la quota en unitats d'emmagatzematge bytes.

Una vegada finalitzada la configuració de quotes, en podeu veure l'estat tant per a les quotes límit com per a les quotes *tally*, amb:

```
ftpquota --show-records --type=limit
ftpquota --show-records –type=tally
```

Per eliminar les quotes dels fitxers *limit* o *tally* feu:

```
ftpquota --delete-record --type=limit --name=usuari –quota-type=user
```

El detall de l'ordre:

- *–delete-record*: elimina el registre del fitxer *limit* o *tally*.
- *–type*: especifica a quin fitxer eliminarà el registre, fitxer *limit* o *tally*.
- *–name*: especifica el nom de l'usuari o grup a qui és aplicable la quota.
- *–quota-type*: especifica si la quota és per usuari o grup, *user* o *group*.

Per actualitzar les quotes del fitxer *limit* o *tally* feu:

```
ftpquota --update-record --type=limit --name=usuari --quota-type=user --bytes-
download=100 --files-download=1 –units=B
```

El detall de l'ordre:

- *–update-record* : actualitza el registre del fitxer *limit* o *tally* amb les configuracions noves.
- *–type*: especifica a quin fitxer actualitzarà el registre, fitxer *limit* o *tally*.
- *–name*: especifica el nom de l'usuari o grup a qui és aplicable la quota.
- *–quota-type*: especifica si la quota és per usuari o grup, *user* o *group*.
- *–bytes-download*: especifica el nombre màxim de bytes que s'han de descarregar de la quota en unitats d'emmagatzematge bytes.
- *–files-download*: especifica el nombre de fitxers que es poden descarregar del servidor.
- *–units*: unitats que es faran servir per calcular les unitats d'emmagatzematge i processament de dades (B o byte, Kb o quilo, Mb o mega i Gb o giga. Per defecte fa servir byte si no es fa servir la directiva *units*.

6. Tipus d'usuaris i accessos al servei

Hi ha dos tipus d'usuaris dins del servei FTP:

- **Usuari del sistema**: usuari propi del sistema on hi ha el servei FTP i que accedeix al seu directori personal.
- **Usuari anònim**: usuari que no té contrasenya de validació i d'accés públic al servei.

D'aquest tipus d'usuaris podem especificar-ne un tercer que deriva dels usuaris de sistema, anomenats **usuaris virtuals**. La diferència ve donada perquè aquests no són dependents del sistema sinó del servidor FTP directament.

Els permisos es poden configurar per usuari i per grups de treball. Adreceu-vos a l'apartat "Permisos" per a més detall.

Tots els usuaris que tenen accés al servei FTP, com a administradors del servei, habilitareu permisos per poder manipular els fitxers i directoris del servei. Aquests permisos poden ser lectura, escriptura, llistar, eliminar, etc. Són els que permeten treballar amb els directoris d'accés realitzats a la configuració del servei i els que permet la definició del protocol.

Dins dels tipus d'usuaris diferenciarem també els tipus d'accessos al servei.

Accés anònim.

Els servidors poden oferir servei lliurement a tots els usuaris, accedir sense tenir un identificador d'usuari, llegir i navegar pel contingut dels directoris lliurement, indiferentment de qui hi accedeix i del lloc on ho fa.

L'accés anònim és una forma còmoda de permetre que tots els clients tinguin accés a certa informació sense que l'administrador del servei hagi de controlar els comptes d'usuaris.

La informació que es treballa per a l'accés anònim és de caràcter públic i es poden llegir els continguts dels directoris però no eliminar-los ni modificar-los. Normalment, el contingut sol ser programari de domini públic o de lliure distribució, imatges, so, vídeos, etc.

Exemples de servidors públics amb accés anònim: ftp://ftp.rediris.es i ftp://cdimage.ubuntu.com.

El requisit per accedir per accés anònim és mitjançant un nom predefinit que existeix en el servei FTP i que ha d'estar configurat prèviament.

Aquest usuari que permet l'accés anònim es diu *anonymous*. Quan es valida la connexió, el nom de l'usuari que posem és *anonymous*, i sense contrasenya (encara que demani una contrasenya, no és necessari escriure res o, si ho demana obligatòriament, podeu posar qualsevol correu electrònic com a contrasenya vàlida).

L'accés anònim és un tipus d'accés que és inviable en el cas del desplegament web, on el control d'accés dels usuaris és important, ja que és de caràcter privat, confidencial i depèn també la nostra aplicació web. Permetre un accés al directori arrel de l'aplicació web amb un accés anònim mitjançant FTP és una falta greu de seguretat i amb conseqüències desastroses.

Accés per usuari identificat

Es dona en els casos de la necessitat de privilegis i la informació amb la qual es treballa és d'indole privada. S'ha d'accedir al servei mitjançant usuaris identificats dins de l'FTP, anomenats comptes.

Els comptes d'usuari poden ser:

- **Usuari de sistema**: usuari definit dins del sistema on s'ofereix el servei.
- **Usuari virtual**: no té una relació directa amb el sistema.

Tots aquests usuaris tindran configurat una sèrie de permisos depenent de la implicació que tinguin els usuaris, per exemple dins del projecte web. Us poden interessar usuaris que només puguin llegir la informació del projecte i d'altres que puguin actualitzar els fitxers, tot això gestionant la jerarquia de l'equip del projecte que està fent l'aplicació web.

7. Creació d'usuaris a ProFTPD

Hi ha dos tipus d'usuari que es poden crear per accedir a ProFTPD:

- Usuari de sistema, creat dins de */etc/shadow* (fitxers d'usuaris) i */etc/passwd* (fitxer de configuració de l'usuari) i grups a */etc/group*.
- Usuari virtual, fitxer per a ProFTPD amb usuari no dependent del sistema.

7.1. Usuari de sistema a ProFTPD

Per crear usuaris de sistema i que tinguen vinculació amb ProFTPD, procediu a fer les comprovacions i executar les ordres detallades.

Modifiqueu el fitxer */etc/shells* amb:

```
sudo nano /etc/shells
```

Afegiu-lo al final de tot de la línea */bin/false* com a resultat:

/bin/false

Si deixeu accés als usuaris amb un *shell* vàlid podran accedir al servei FTP dins del directori arrel configurat o al seu directori *home*.

Quan afegiu /bin/false permetrà que quan es creïn usuaris hi afegiu el *shell /bin/false*, així evitareu que els usuaris FTP tinguin accés al servei FTP només si l'administrador permet l'accés.

Creeu un usuari amb l'ordre:

```
useradd usuari -d directoriTreball -s /bin/false
passwd usuari
```

On:

- usuari: nom de l'usuari que creareu dins del sistema.
- directoriTreball: directori de treball de l'usuari, per exemple */home/usuari*.

Eliminació de l'usuari:

```
sudo userdel -r nomUsuari
```

Creació del grup de treball:

```
sudo groupadd nomGrup
```

Eliminació del grup de treball:

```
sudo groupdel nomGrup
```

7.2. Usuari virtual a ProFTPD

Per crear usuaris virtuals dins de ProFTPD, executeu l'ordre *ftpasswd*, que crearà un fitxer anomenat *ftpd.passwd*:

sudo mkdir /etc/proftpd/usuaris
cd /etc/proftpd/usuaris
sudo ftpasswd --passwd --name=nomUsuari --home=pathTreball --shell=/bin/false --uid=500

On:

- *–name*: nom de l'usuari que creareu.
- *–home*: directori de treball, per exemple /home/usuari.

Per crear grups de treball virtuals dins de ProFTPD, executeu l'ordre *ftpasswd* que crearà un fitxer anomenat *ftpd.group*:

cd /etc/proftpd/usuaris
sudo ftpasswd --group --name=nomGrup --gid=idGrup --member=nomUsuari

On:

- *–name*: nom del grup que creareu.
- *–gid*: id del grup en format numèric enter.

Dins del fitxer de configuració *proftpd.conf*, afegiu les directives que permeten que s'agafin els usuaris i grups virtuals creats.

RequireValidShell off
AuthUserFile /etc/proftpd/ftpd.passwd
AuthGroupFile /etc/proftpd/ftpd.group

8. Modes de connexió del client

El mode de transferència s'estableix a l'inici de les comunicacions FTP i és dependent de com és el sistema de fitxers del servidor.

La majoria dels sistemes fan ús de l'estructura de fitxers binaris, antics sistemes Unix i *mainframe* i poden utilitzar l'estructura de fitxers ASCII. En qualsevol cas, aquest mode es

decideix dins del servidor FTP i el client automàticament detectarà quin dels dos modes farà servir.

El protocol FTP es basa en el protocol TCP, amb dos canals de dades amb diferents ports, un per enviar les dades i l'altre per enviar les ordres.

Els ports que es fan servir per defecte són:

- Port número 21, per al canal de control
- Port número 20, per al canal de dades de transmissió

Dins del protocol FTP es defineixen dos modes de connexió que es configuren dins del servei, el mode ftp actiu i el mode ftp passiu.

Mode FTP actiu

El mode actiu de FTP és un protocol de comunicació utilitzat per transferir fitxers entre un client i un servidor a través de la xarxa. En aquest mode, el client FTP inicia la connexió de dades amb el servidor, la qual cosa significa que és el client qui estableix la connexió amb el servidor per transferir fitxers.

Això funciona de la següent manera:

1. El client FTP inicia una connexió de control amb el servidor FTP mitjançant el port 21. A través d'aquesta connexió, el client envia comandes al servidor per gestionar les operacions de transferència de fitxers, com ara llistar directoris o demanar fitxers.
2. Quan el client vol transferir un fitxer, envia una sol·licitud al servidor per obrir una connexió de dades mitjançant un port específic (anomenat port d'ordre o port de dades) en el servidor. Aquest port d'ordre és un número aleatori assignat pel client.
3. El servidor FTP, en mode actiu, obre una connexió de dades amb el client. Això significa que el servidor actua com a client en aquesta nova connexió, establint una connexió d'eixida cap al port especificat pel client. A través d'aquesta nova connexió de dades, es realitza la transferència del fitxer.
4. Un cop finalitzada la transferència, la connexió de dades es tanca i el client i el servidor continuen comunicant-se a través de la connexió de control.

Aquesta és la diferència clau entre el mode actiu i el mode passiu de FTP. En el mode passiu, és el servidor qui inicia la connexió de dades cap al client, el que pot ser útil quan el client està darrere d'un tallafocs i no pot acceptar connexions entrants.

Mode FTP passiu

Per evitar que el servidor inicie la connexió al client hi ha un altre mètode de connexió anomenat passiu.

Aquest mode és una opció alternativa al mode actiu de FTP i és útil en situacions en les quals el client o el servidor estan darrere d'un tallafocs o tenen restriccions en l'establiment de connexions de dades d'eixida.

El mode passiu funciona de la següent manera:

1. El client FTP inicia una connexió de control amb el servidor FTP mitjançant el port 21, tal com ho faria en el mode actiu.
2. Quan el client vol transferir un fitxer, en lloc de demanar al servidor que obri una connexió de dades mitjançant un port específic com en el mode actiu, el client demana al servidor que siga ell qui obri una connexió de dades passiva. El servidor FTP respondrà indicant un port i una adreça IP a través de la qual el client pot connectar-se.
3. El client FTP, en mode passiu, estableix una nova connexió de dades cap al port i l'adreça IP proporcionats pel servidor. Això permet la transferència de fitxers sense problemes, ja que el client està fent la connexió de dades eixint cap al servidor.
4. Una vegada finalitzada la transferència, la connexió de dades es tanca i el client i el servidor continuen comunicant-se a través de la connexió de control.

El mode passiu és útil quan el client o el servidor estan en una xarxa protegida per un tallafocs, ja que les connexions de dades sortints són més fàcils d'acceptar que les connexions entrants. Aquest mode ajuda a superar aquesta limitació i permet una transferència de fitxers més eficaç.

Amb el mode passiu es resolen molts problemes del client, però s'amplien els problemes del servidor. Un dels principals problemes és l'obertura d'un gran rang de ports en el servidor per poder iniciar canals de dades.

Un dels avantatges actualment és que les implementacions de servidors FTP permeten escollir el rang de ports que es faran servir.

Per realitzar la configuració dins de ProFTPD en el mode passiu, afegiu dins de la configuració del fitxer */etc/proftpd/proftpd.conf* la directiva:

```
PassivePorts 1024 2000
```

El rang de ports de configuració anirà mínim del port número 1024 fins, com a màxim, al 65536.

9. Protocol de transferència de fitxers segur

Quan es va redactar el protocol FTP dins la RFC 959 la seguretat no era un tema crític. Amb l'evolució de les xarxes i la transferència massiva de dades dins les xarxes públiques, ha canviat molt respecte a les idees originals en els anys 70 i 80. Aquesta evolució fa que enviar dades sense encriptar siga molt arriscat i que protocols antics haja d'evolucionar per garantir que la tramesa de dades siga segura.

Amb l'evolució de les xarxes, el protocol FTP va patir noves revisions per pal·liar les deficiències de seguretat i l'any 1997 es va redactar l'actualització del protocol FTP que dóna com a resultat l'FTPS.

Els autors de la RFC van llistar l'any 1999 les diferents vulnerabilitats FTP:

- Atacs Spoofing
- Atacs de força bruta
- Atacs rebot (*bounce attacks*)
- Captura de paquets (*sniffing*)
- Robatori de ports (*port stealing*)
- Claus d'usuari i dades no xifrades

El protocol FTPS fa ús d'SSL, TLS i és una combinació d'algoritmes de xifrat asimètrics (RSA,DSA), algoritmes simètrics (DES/3DES,AES etc.) i un algoritme d'intercanvi de claus amb autenticació de certificats X.509.

Hi ha dos modes de treball amb FTPS:

- El **mode FTPS implícit SSL**. Es requereix una sessió SSL entre el client i el servidor abans que s'intercanviï qualsevol dada. Com el nom diu, l'ús d'SSL és implícit i qualsevol intent de connexió dels clients sense fer ús d'SSL és rebutjat pel servidor. Els ports de treball de l'FTPS implícit són el 990 i el 989.

Actualment es fa servir molt poc FPS implícit a favor de fer ús del segon mètode FTPS explícit SSL.

- El **mode FTPS explícit SSL**. El client i el servidor negocien el nivell de protecció que es farà servir. Aquesta situació és útil per poder treballar amb el mateix port amb sessions encriptades o no encriptades.

En el mode explícit SSL el client inicia una connexió sense encriptar amb el servidor FTP. El client demana una petició al servidor FTP per iniciar una ordre sobre SSL, enviant les ordres AUTH TLS o AUTH SSL.

Una vegada iniciat el canal SSL el client envia les credencials al servidor FTP. Totes les credencials són encriptades i enviades pel canal SSL. De la mateixa manera que s'ha fet la protecció del canal de control, el canal de dades segueix el mateix procediment fent ús de l'ordre PROT. Els ports que fa servir són el 20 i el 21, on s'efectua el xifrat.

Hi ha una altre tipus de servei segur amb FTP anomenat **SFTP**.

SFTP sol ser confós amb el servei FTPS i viceversa, i realment no tenen res a veure mútuament. Excepte per la seguretat en la tramesa de fitxers, el procediment intern és diferent.

SFTP està basat en SSH (*secure shell*), protocol conegut per proveir seguretat als terminals remots.

No fa ús de canals d'ordres i de dades. Els dos canals que fem servir en FTPS s'envien en paquets amb format dins d'un mateix canal, és a dir, el canal de dades i d'ordres és únic.

Totes les dades enviades i rebudes són encriptades mitjançant un algoritme d'encriptació prèviament acordat.

Les sessions estan protegides mitjançant claus públiques i privades, que ofereixen un sistema d'autenticació conegut com a autenticació de clau pública que es pot fer servir com a alternativa o unió dels sistemes d'autenticació tradicionals de noms d'usuari i contrasenyes.

9.1 Configuració FTPS a ProFTPD

Per configurar FTPS dins de ProFTPD, feu les instal·lacions i configuracions necessàries per habilitar-ho.

Dins dels sistemes Debian o basats en Debian ja pot venir instal·lada l'aplicació *openssl*. Comproveu-ho amb:

```
dpkg -l | grep openssl
```

Procediu a instal·lar el paquet d'eines d'administració i biblioteques relacionades amb encriptació openSSL.

```
sudo apt-get update
sudo apt-get install openssl
```

Creeu el directori */etc/proftpd/ssl* i situeu-vos dins:

```
sudo mkdir /etc/proftpd/ssl
cd /etc/proftpd/ssl
```

Dins del directori */etc/proftpd/ssl* creeu els certificats x.509 per configurar l'FTPS. Executeu:

```
sudo openssl req -new -x509 -days 365 -nodes -out
/etc/proftpd/ssl/proftpd.cert.pem -keyout /etc/proftpd/ssl/proftpd.key.pem
```

Ompliu les dades del certificat amb les dades que us demanarà.

Una vegada finalitzada l'ordre *openssl*, obtindreu dos fitxers:

- Certificat x509, anomenat *proftpd.cert.pem*
- Claus públiques i privades *proftpd.key.pem*

Canvieu els permisos als fitxers de certificats amb permisos de lectura i escriptura per a usuaris.

```
sudo chmod 600 /etc/proftpd/ssl/proftpd.*
```

Un cop obtinguts els certificats, editeu el fitxer de configuració */etc/profptd/tls.conf*:

```
sudo nano /etc/proftpd/tls.conf
```

I afegiu-hi:

```
<IfModule mod_tls.c>
    # Fem servir el servei FTPS
     TLSEngine                  on
   # configuració del log de sortida al path corresponent
```

```
  TLSLog                      /var/log/proftpd/tls.log
     # protocols que es poden fer servir SSLv23 SSLv3 TLSv1 TLSv1.1 TLSv1.2
  TLSProtocol TLSv1.2
     #combinació d'alogritmes per autenticar, xifrar etc.
    TLSCipherSuite AES128+EECDH:AES128+EDH
  TLSOptions                  NoCertRequest AllowClientRenegotiations
      # path dels certificats i claus generades
       TLSRSACertificateFile      /etc/proftpd/ssl/proftpd.cert.pem
    TLSRSACertificateKeyFile   /etc/proftpd/ssl/proftpd.key.pem
     # demana un certificat al client
        TLSVerifyClient            off
  # obliga el client a fer una connexió TLS
       TLSRequired               on
        RequireValidShell         no
</IfModule>
```

Dins del fitxer de configuració proftpd.conf:

```
sudo nano /etc/proftpd/proftpd.conf
```

Afegiu la directiva include */etc/proftpd/tls.conf* que permetrà agafar la configuració per a FTPS.

10. Utilització d'eines gràfiques

Les aplicacions gràfiques que implementen el protocol FTP fan ús dels avantatges dels llenguatges de programació mitjançant les llibreries gràfiques del sistema operatiu i les llibreries que implementen el protocol FTP.

Un exemple de llibreria FTP és la implementació ftplib a Python, que té les funcionalitats necessàries per realitzar un client FTP. Es podria desenvolupar en poc temps un client gràfic o un terminal alfanumèric que faria les mateixes funcionalitats que qualsevol programi que hi ha al mercat.

10.1. Client gràfic Filezilla

Tenim diferents opcions per fer servir clients FTP en mode gràfic. Un dels més coneguts és Filezilla client: filezilla-project.org.

El projecte Filezilla, amb llicència GNU, posseeix una solució FTP com a client per a sistemes Microsoft, Linux i Mac OS i una solució servidor només implementat per sistemes Microsoft.

Característiques principals del client Filezilla:

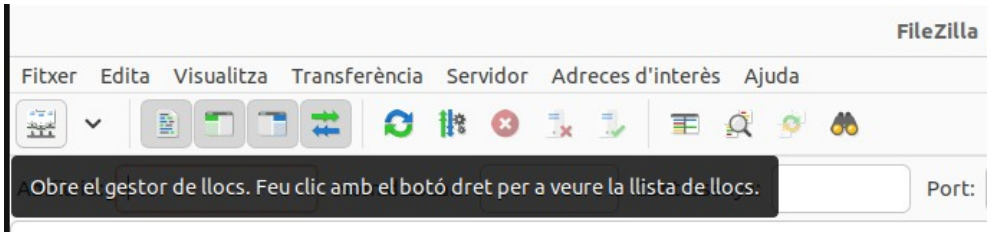
- Suport per a FTP, FTP amb SSL/TLS (FTPS) i SSH FTP (conegur com SFTP)
- Traducció a múltiples llenguatges
- Suport per treballar amb fitxers més grans de 4 GB (GigaBytes)
- Interfície amb pestanyes
- Cua de transferència i gestor d'administració avançat
- Suport per a *Drag & Drop*
- Configuració de velocitats de transferència
- Navegació de directoris sincronitzada

- Cerca de fitxers remot
- Editor de fitxers remot
- Comparació de directoris

Per instal·lar el client Filezilla en el sistema basat en Debian, executeu dins d'un terminal:

```
sudo apt-get update
sudo apt-get install filezilla
```

Executeu Filezilla. Podeu veure que l'entorn gràfic és intuïtiu. Si feu clic en la icona de servidors, baix de fitxers, dins del menú que us mostra permet crear diferents configuracions d'accés a servidors remots FTP.



10.2. Client gràfic dels sistemes operatius

Dins de l'entorn gràfic del sistema operatiu Windows o Linux permeten accedir amb l'explorador de fitxers del sistema a un servidor remot FTP.

Dins del navegador de fitxers dels sistemes Windows fiquem com a ruta de navegació ftp://cdimage.ubuntu.com. Veureu que farà la connexió que us enllaça al servidor FTP remot. Pot ser que ens demane l'usuari i la contrasenya si és necessari.

Per poder copiar o escriure fitxers dins d'una connexió FTP treballeu de la mateixa manera que si fos una carpeta local del sistema o en xarxa. Això sí, no podreu modificar els continguts si esteu en una connexió anònima o l'administrador no ho permet.

En el cas del funcionament dins d'un sistema Linux serà exactament igual que en Windows, però ara dins del navegador de fitxers del sistema Linux.

10.3. Utilització del servei de transferència de fitxers des del navegador

Un dels grans avantatges que té el protocol FTP es que permet ser implementat fàcilment en diferents tipus de sistemes operatius (escriptori, mòbils, consoles, etc.).

A més, el protocol FTP és incorporat dins d'aplicacions, com una extensió de la funcionalitat principal de l'aplicació. Imagineu aplicacions IDE de desenvolupament o els navegadors web.

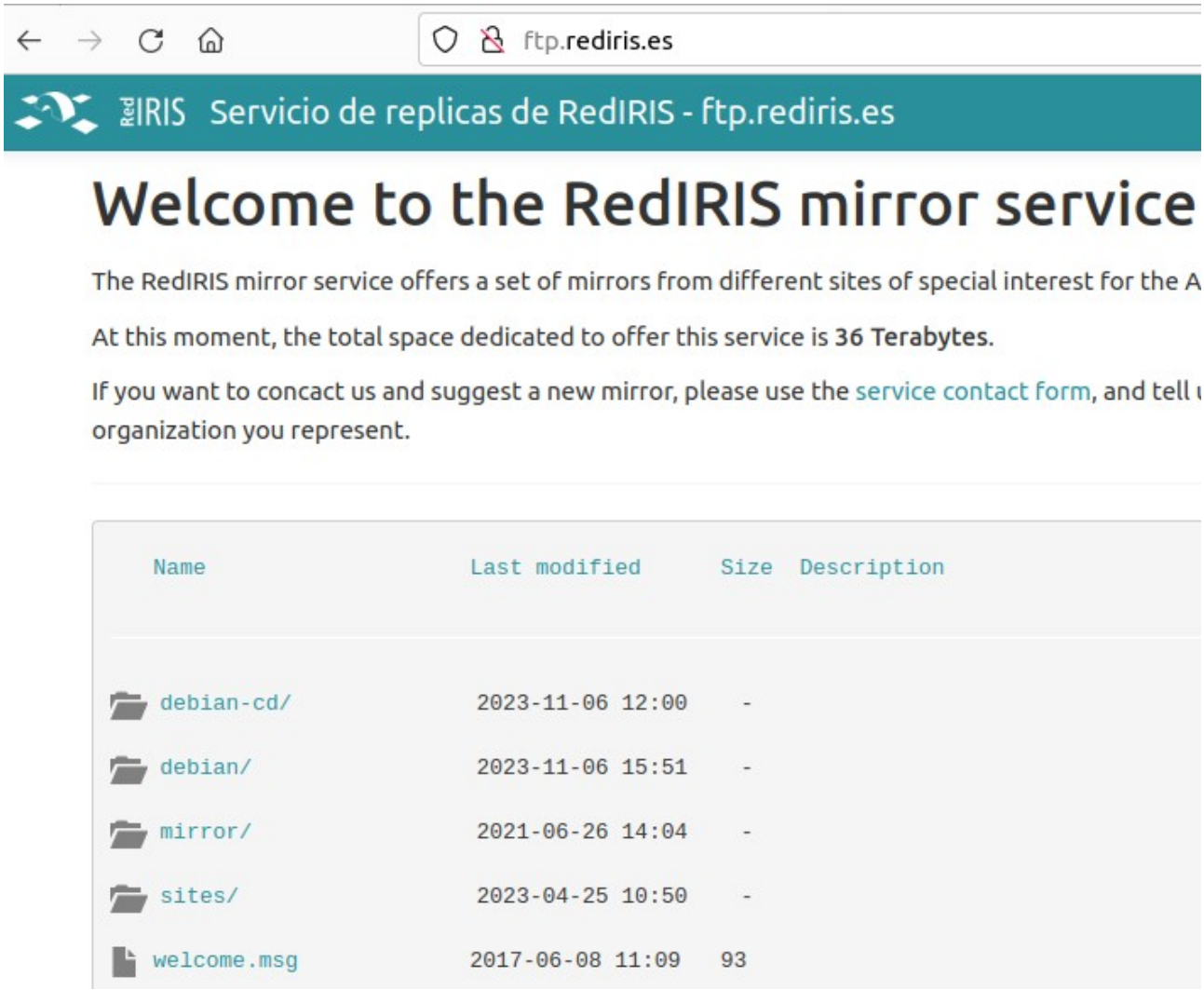
Els navegadors web permeten accedir als servidors FTP com un client FTP directe, amb la seva validació d'usuari, llegir el directori remot i accedir a la informació.

La forma d'accedir a un servidor FTP mitjançant el navegador és escriure dins de la barra de navegació l'esquema:

```
ftp://<url>
```

Vegeu un exemple bàsic dins del navegador. Obriu una finestra i copieu a la barra de navegació la següent direcció: ftp://ftp.rediris.es.

Veureu que dins de la part de visualització web es llista un directori amb el seu contingut. Aquesta llista del directori pertany a l'arrel del servidor remot al qual ens hem conectat. Proveu de navegar i de baixar algun contingut.



La limitació que ofereix el navegador és que no podem modificar els fitxers. Això vol dir que les accions crear, modificar o esborrar fitxers no es poden realitzar. Per aquesta deficiència, hi ha una altra alternativa, que és la d'ampliar el funcionament del navegador amb extensions del navegador.

Les extensions no són més que programari alternatiu que funcionen juntament amb el navegador per ampliar les seves capacitats. Feu una cerca dins de Google Chrome o Mozilla Firefox de les diferents extensions que es poden afegir al navegador.

11. Utilització del servei de transferència de fitxers en el procés de desplegament de l'aplicació web

El procés de desplegament (en anglès *deployment*) consisteix en l'encapsulació del projecte i la distribució final dins d'un entorn de producció on s'executarà.

La distribució final de l'aplicació web a un entorn de producció es pot dur a terme:

- Mitjançant un fitxer d'encapsulació que engloba tot el contingut del projecte i que depèn de la tecnologia que fem servir per desenvolupar l'aplicació web.
- Mitjançant una estructura de directoris. Aquests directoris contenen els fitxers html, php, fulles d'estil i fitxers com imatges, sons, vídeos, etc.

La pràctica habitual és dissenyar l'aplicació web en un disc dur en local amb un servidor web local (anomenat entorn de desenvolupament), per després enviar-ho a un servidor final de producció. A vegades existeix un pas intermedi que consisteix a enviar l'aplicació, en un entorn de proves el més semblant possible a l'entorn de producció.

La transferència dels fitxers de l'aplicació web es farà mitjançant un client FTP cap a un servidor FTP que enllaçarà amb el directori principal del servidor web. Aquesta transferència de fitxers es farà mitjançant un usuari autenticat dins del servidor FTP.

Les dades necessàries per fer la transferència seran:

- URL o adreça IP del servidor.
- Usuari i contrasenya creats dins del servidor FTP i amb permisos d'escriptura als directoris on s'emmagatzemen els fitxers de l'aplicació web.

El client FTP que feu servir per a pujar l'aplicació és indiferent, podeu fer servir clients FTP integrats en els IDE de desenvolupament, clients FTP dins del terminal Unix, Microsoft o clients gràfics.

El desplegament de l'aplicació web, dins del cas d'actualització de les funcionalitats, sempre serà en hores on l'ús del servei és minoritari, per evitar problemes als usuaris. A vegades també l'aplicació web pot tenir programada una opció per aturar el servei i permetre que es puguen actualitzar els continguts del servidor.

Índex:

1. Introducció

Actualment, per a desenvolupar una aplicació cal conèixer i utilitzar múltiples tecnologies. No només és necessari dominar el llenguatge en què es desenvoluparà l'aplicació, sinó que també cal fer servir un conjunt d'eines per facilitar l'assemblatge, el manteniment i el desplegament de l'aplicació.

Aquestes eines permeten configurar l'entorn de desenvolupament per treballar en múltiples projectes al mateix temps, encara que els requisits del sistema de desplegament siguen molt diferents de les característiques del vostre equip. També permeten crear contenidors per desplegar les aplicacions web en entorns controlats, optimitzar el procés de generació dels fitxers que formen l'aplicació i automatitzar tasques per simplificar l'assemblatge.

En aquesta unitat aprendrem a crear entorns de desenvolupament amb Vagrant i contenidors amb Docker, gestionar un servei LDAP mitjançant l'eina phpLDAPAdmin, per a què serveixen els servidors de DNS, i com utilitzar eines basades en Node.js per millorar el flux de treball utilitzant preprocessadors de CSS, compiladors de JavaScript i automatitzadors de tasques.

A l'apartat **"Serveis de xarxa"** es descriu què són els contenidors i en què consisteix la virtualització. Seguidament, es detalla el funcionament de Docker, incloent un exemple pas a pas de com crear i desplegar un contenidor amb Docker. A continuació, s'explica què és Vagrant i com instal·lar i utilitzar una caixa de Vagrant al vostre equip. Finalment, es parla del servei de noms (DNS) i els serveis de directoris (LDAP).

A l'apartat **"Utilització de serveis de xarxa i automatització"** s'ensenya a utilitzar Node.js i el gestor de paquets npm per instal·lar eines per a la línia d'ordres basades en Node.js, com ara el preprocessador de CSS (Sass, Less i stylus), el compilador d'ES5 Babel (per ES2015 i TypeScript) i les eines d'automatització Gulp i Grunt.

Per assimilar correctament els coneixements que comprenen aquesta unitat, és molt important instal·lar el programari indicat al vostre equip i seguir els exemples mostrats pas a pas.

A l'hora de desenvolupar una aplicació web moderna s'han de tenir en compte aspectes que van més enllà de la programació de la mateixa aplicació. Cal tenir en compte que el sistema en què es desplega l'aplicació és diferent de l'entorn de desenvolupament i, molt sovint, això provoca problemes que no es detecten fins al moment del desplegament, és a dir, massa tard.

Per evitar aquesta situació, es poden utilitzar dues tecnologies molt relacionades entre si:

- **Contenidors**: la tecnologia de contenidors permet utilitzar un sistema d'imatges per desplegar exactament les mateixes configuracions en múltiples màquines, utilitzant el nucli del sistema operatiu de la màquina amfitriona i sense que afecte al rendiment. D'aquesta manera, es treballa sobre la configuració que es troba a la màquina de desplegament.
- **Virtualització**: aquesta tecnologia permet crear màquines virtuals amb configuracions similars a la de la màquina –o màquines– de desplegament per emular les condicions en què ha de funcionar l'aplicació.

En el cas de la tecnologia de contenidors, la implementació més popular és **Docker**, que té col·laboradors com Google, IBM o Cisco. Per instal·lar-lo només es necessita una màquina amb Linux i la instal·lació de Docker.

Les característiques més importants de Docker són les següents:

- Fa servir el nucli de Linux i, per conseqüent, no es pot fer servir amb altres sistemes operatius.
- Utilitza un repositori d'imatges que s'utilitzen com a base per als nous contenidors.
- Fa servir un model de capes (al fitxer Dockerfile) per construir la imatge que, en executar-se, posa en marxa el contenidor.

Pel que fa a la virtualització, l'aplicació més utilitzada és **Vagrant**. Aquesta tecnologia permet utilitzar diferents eines de virtualització, com Virtual Box o VMware, per crear entorns de desenvolupament. D'aquesta manera, es pot treballar localment amb diferents versions de sistemes operatius, servidors web, llenguatges, etc., sense haver de canviar d'eines ni d'equip.

Per sincronitzar les carpetes on es troben els fitxers amb el codi de l'aplicació per desenvolupar i les carpetes de l'entorn virtualitzat, es fa servir el sistema de carpetes compartides proporcionat per les eines de virtualització. Així, per exemple, mentre que la màquina de desplegament pot córrer amb Windows 10 sense tenir instal·lat cap servidor web ni cap versió de PHP, a l'entorn de desplegament pot executar-se Ubuntu amb Apache i PHP. A l'hora de provar l'aplicació, es pot visualitzar en un navegador introduint-hi l'adreça IP de la màquina virtual. També s'hi pot accedir mitjançant SSH des de la línia d'ordres.

Per instal·lar Vagrant cal descarregar prèviament un proveïdor (el recomanat és VirtualBox). Es pot trobar un instal·lador de Vagrant per a les plataformes Windows, Linux i macOS.

Les característiques més importants de Vagrant són les següents:

- Fa servir un sistema de caixes per configurar els entorns.
- Permet que tots els membres d'un equip facin servir un entorn idèntic de desenvolupament.
- Utilitza un sistema de carpetes compartides per sincronitzar la informació entre l'equip i l'entorn virtualitzat.
- Permet emular conjunts de màquines.

El **servei de sistema de noms de domini (DNS)** és el responsable de convertir els noms de domini (per exemple, ioc.xtec.cat) en l'adreça IP corresponent (per exemple, 83.247.151.178).

Tot equip connectat a internet té una adreça IP que l'identifica. Aquesta adreça pot estar formada per 4 nombres de 8 bits (protocol IPv4) o 8 nombres de 16 bits (protocol IPv6), però no totes corresponen a un nom de domini concret. A més a més, no totes les adreces són públiques, ja que hi ha uns intervals d'adreces que són utilitzats per a la configuració de xarxes privades.

El sistema de noms de domini consisteix en una estructura de dades en forma d'arbre, subdividit en zones associades a un o més dominis o subdominis. Segons aquesta jerarquia, cada element que forma part d'un domini separats per un punt es considera un subdomini del domini de primer nivell (l'element més a la dreta).

Per resoldre una consulta a un servidor DNS, es connecta amb el servidor de DNS (configurat per l'administrador del sistema o proporcionat pel proveïdor d'internet), i aquest fa la consulta a un servidor més específic. Si no es troba, s'envia la consulta al següent servidor més específic, i aquest procés es repeteix fins a obtenir la resposta o un missatge d'error.

El programari per a servidors DNS més utilitzat a internet és BIND (Bekeley Internet Name Domain), però a xarxes locals és més habitual trobar el servidor DNS de Windows.

Els diferents tipus de servidors que formen un servei DNS són els següents:

- **Servidors de noms cau**: emmagatzemen les adreces resoltes per accelerar consultes futures.
- **Servidors mestres**: gestionen els fitxers de zona i notifiquen quan s'han produït canvis als servidors esclaus.
- **Servidors esclaus**: es connecten amb el servidor mestre per rebre actualitzacions.

Els **fitxers de zona** contenen tota la informació necessària per resoldre les consultes als servidors DNS.

Els **serveis de directori** poden comparar-se a una guia de telèfons on s'enllacen els recursos d'una xarxa amb les seves adreces, i permeten gestionar, administrar i organitzar els seus elements.

LDAP es pot configurar i modificar utilitzant fitxers, però aquest sistema no és gens amigable. Per aquesta raó és preferible utilitzar interfícies gràfiques, per exemple, mitjançant un navegador. Una de les eines que permet gestionar **LDAP** mitjançant el navegador és **phpLDAPadmin**. Com que hi ha grans organitzacions que utilitzen LDAP o **AD** (**Active Directory**, la implementació d'LDAP per a Microsoft Windows), existeixen molts llenguatges que ofereixen biblioteques per accedir a la informació que subministra LDAP.

El desenvolupament de les aplicacions web modernes implica la utilització de múltiples tecnologies que permeten simplificar i automatitzar les tasques.

La base per a la majoria d'aquestes eines es **Node.js**, una implementació de JavaScript que pot executar-se al servidor. Això permet desenvolupar serveis web síncrons (per exemple, servidors de xat) i asíncrons (per exemple, servidors web), així com eines per a la línia d'ordres.

Juntament amb la instal·lació de Node.js hi ha **npm**, un gestor de paquets que permet instal·lar nous mòduls i eines basades en Node.js. Aquest és el gestor de paquets més utilitzat en el desenvolupament d'aplicacions web. Per altra banda, **Yarn** és un gestor de paquets desenvolupat per Facebook que pot ajudar a solucionar determinats problemes en aplicacions molt grans.

Els preprocessadors de CSS permeten generar fitxers CSS utilitzant altres llenguatges (Sass, Less i Stylus) per afegir noves característiques, com són la utilització de variables, funcions i herència, de manera que se simplifica la codificació d'aquests fitxers.

L'inconvenient és que aquests fitxers s'han de preprocessar per generar els fitxers CSS cada cop que es produeix un canvi.

Els tres preprocessadors de CSS més populars són:

- **Sass**: és un dels primers preprocessadors de CSS.
- **Less**: és menys potent que Sass però més senzill d'aprendre.
- **Stylus**: és el més recent dels tres. Inclou la majoria de les opcions de Sass i Less, a més de les pròpies.

Tot i que el llenguatge que entenen els navegadors és JavaScript 5, també conegut com a **ES5** (les versions més recents encara no són admeses completament per cap navegador), és possible escriure les aplicacions fent servir ES2015 o en altres llenguatges compilables a ES5 (com TypeScript).

Per convertir aquestes aplicacions, es fan servir compiladors que converteixen el codi a ES5, que fa que sigui interpretable per la majoria dels navegadors. El més popular d'aquests compiladors és Babel.

Els llenguatges que són compilats més habitualment són:

- **ES2015** i posteriors: és la versió més recent de JavaScript, però encara no és admesa per tots els navegadors. Això permet als desenvolupadors escriure el codi amb l'última versió aprofitant totes les millores afegides, de manera que en el futur, quan es pugui fer servir directament, no s'hi hagi de reescriure el codi per aprofitar aquests avantatges.
- **TypeScript**: és un llenguatge desenvolupat per Microsoft basat en l'especificació d'ES2015 a la qual s'afegeixen diverses característiques com la tipificació estàtica.

Com que en el desplegament d'una aplicació cal fer moltes operacions diferents i repetitives, és fàcil cometre errors. Per solucionar aquest problema es fan servir eines d'automatització com **Gulp** i **Grunt**, que permeten automatitzar molts dels processos necessaris per assemblar l'aplicació:

- Preprocessar els fitxers Sass, Less i Stylus.
- Concatenar els fitxers CSS i JS per reduir el nombre de peticions HTML.
- Compilar els fitxers ES2015 o TypeScript en el format ES5.
- Copiar els fitxers necessaris d'unes carpetes a altres (per exemple, les fonts o imatges de les biblioteques a les carpetes públiques de l'aplicació).

La diferència més important entre els automatitzadors Gulp i Grunt és que Gulp es configura mitjançant codi JavaScript, mentre que Gulp utilitza fitxers de configuració.

2. Serveis de xarxa

A l'hora de desenvolupar una aplicació web, s'ha de tenir en compte que l'entorn de desplegament no és el mateix que el de desenvolupament. Aquestes diferències acostumen a provocar errors, en desplegar l'aplicació, que no es produeixen durant el desenvolupament local. Per resoldre aquests problemes s'acostuma a recórrer a dues

tècniques diferents, que poden utilitzar-se combinades: la utilització de contenidors i la virtualització.

Els **contenidors** permeten configurar els entorns de desplegament de manera que poden reproduir-se de forma idèntica en qualsevol màquina, independentment del sistema operatiu i la configuració de l'amfitrió. D'aquesta manera, es pot fer servir la mateixa configuració per al servidor de desplegament, el servidor de proves i els equips locals de desenvolupament.

La **virtualització** consisteix a crear una màquina virtual amb una configuració semblant a la del servidor de desplegament. Per facilitar aquesta tasca, hi ha programes com Vagrant, que permeten configurar el seu entorn de desenvolupament de la mateixa manera a un mateix equip de programadors, ja que permet crear les màquines virtuals i aprovisionar-les a partir d'un mateix fitxer de configuració en lloc de fer les instal·lacions individualment.

Pel que fa als serveis de xarxes, un dels més utilitzats és el servei de DNS, ja que es fa servir tant a internet com en xarxes privades. Aquest servei permet traduir un nom de domini en una adreça IP, i a la inversa. Així és com el navegador detecta que en accedir a https://google.es cal connectar amb el servidor que es troba a la IP 216.58.211.195.

Entre les implementacions de servei de directori més utilitzades hi ha l'LDAP. Aquest servei permet consultar informació sobre usuaris i recursos de la xarxa, així com autenticar els usuaris (fins i tot en altres serveis de tercers). LDAP pot gestionar-se mitjançant fitxers des de la línia d'ordres, però s'acostumen a utilitzar interfícies gràfiques com phpLDAPAdmin, un client LDAP al qual s'accedeix des del navegador.

2.1. Contenidors i virtualització

Cal tenir en compte que a l'hora de desenvolupar i desplegar una aplicació web és habitual haver de treballar en diferents entorns (configuracions de maquinari i programari). Per una banda, hi ha l'entorn de desplegament, que és on es desenvolupa l'aplicació (probablement l'ordinador personal) i, per l'altra, hi ha l'entorn de proves on es comprova que l'aplicació funciona correctament abans de fer-ne el desplegament (aquest entorn pot coincidir amb el de desenvolupament o trobar-se en una altra màquina en una xarxa local o remota). Finalment, hi ha l'entorn de producció, que es troba en un servidor remot.

Com que cadascun d'aquests entorns pot tenir una configuració de maquinari i programari diferent, sovint hi ha problemes en desplegar aplicacions web: mentre que a l'entorn de desenvolupament pot funcionar perfectament, en desplegar-la a qualsevol dels altres entorns poden produir-se errors provocats per les diferències entre sistemes operatius, servidors webs, biblioteques instal·lades o, fins i tot, pel sistema de fitxers.

Per minimitzar aquests problemes es pot recórrer a la utilització de sistemes de contenidors (i així ens assegurem que la configuració de l'entorn de proves i el de desplegament és idèntica) i a la virtualització (per crear entorns de desenvolupaments el més similar possibles a l'entorn de producció o desplegament).

Tot i que la utilització de **contenidors i màquines virtuals** pot semblar molt similar, no es tracta del mateix concepte.

- Els **contenidors** són molt més lleugers i pràcticament no afecten el rendiment de l'aplicació, fet que possibilita utilitzar-los en el desplegament.
- Les **màquines virtuals** són instal·lacions completes del sistema operatiu i afegeixen capes extres a l'execució dels programes, ja que cada instrucció ha de passar pel sistema operatiu virtualitzat, el programari de virtualització, el maquinari de virtualització de l'equip hoste i el nucli del sistema operatiu hoste.

Els **contenidors** utilitzen el mateix nucli del sistema operatiu i, per aquest motiu, no hi ha cap disminució apreciable de rendiment. A més a més, l'espai que ocupa en disc és molt reduït, ja que només s'hi han d'afegir els fitxers específics que requereix l'aplicació que s'executa en el contenidor.

En el cas de sistemes distribuïts (una aplicació desplegada a múltiples màquines), és molt útil utilitzar contenidors, ja que només cal preparar el contenidor una vegada i instal·lar-lo en tants servidors com siga necessari. Aquest és un dels motius pels quals Google fa servir contenidors per desplegar les seves aplicacions en lloc d'haver de configurar milers d'equips individualment.

Un inconvenient d'aquesta tecnologia és que no és possible fer servir contenidors que utilitzen un sistema operatiu en un altre de diferent (per exemple, un contenidor de Linux a Windows). Per aquesta raó, en entorns de desenvolupament s'ha de recórrer a la virtualització.

Docker és un sistema de contenidors amb llicència de programari lliure.

Tot i que aquesta tecnologia es troba disponible des dels anys 80 al sistema operatiu UNIX, fins a l'aparició de Docker, l'any 2013, no era apenes popular. Actualment és fàcil trobar proveïdors de serveis d'allotjament que admeten Docker i permeten fer el desplegament de les aplicacions automàticament (per exemple, Amazon Web Services i Google Cloud).

Contenidors en entorns virtualitzats

En cas d'haver de treballar amb contenidors en ordinadors amb un sistema operatiu diferent del del contenidor, es pot recórrer a la utilització de màquines virtuals. Cal tenir en compte que es perd part de l'eficiència proporcionada per aquesta tecnologia, però és habitual treballar d'aquesta manera en entorns de desenvolupament, ja que l'equip de programadors pot treballar amb el sistema operatiu que s'adapte més bé a les seves necessitats. A més a més, en aquests casos, la pèrdua de rendiment no és crítica.

En el cas de les **màquines virtuals**, cada màquina es troba completament aïllada de la màquina hoste i, per consegüent, això afecta el rendiment: cada acció ha de ser processada per la màquina virtual, el programari de virtualització i la màquina hoste. A més, la instal·lació del programari de cada màquina ha de ser completa, és a dir, ha d'incloure com a mínim tots els fitxers del sistema operatiu per funcionar i, per tant, l'espai en disc necessari serà molt més gran que en el cas dels contenidors.

Per altra banda, com que es tracta d'una instal·lació completa, és possible fer servir màquines virtuals amb diferents sistemes operatius independentment del que estiga instal·lat a la màquina hoste (per exemple, una màquina virtual amb Linux a una màquina hoste amb Windows).

Cal destacar VirtualBox i VMware com a programaris de virtualització perquè són els més populars i perquè ofereixen versions gratuïtes. Tots dos poden utilitzar-se per realitzar instal·lacions completes d'entorns de desenvolupament. Això sí: en lloc d'accedir directament a la màquina creada, s'acostuma a utilitzar altres programes com Vagran, que gestionen el programari de virtualització per generar aquests entorns, automatitzant part de les tasques i simplificant moltes accions habituals, com pot ser la creació d'una màquina base a partir d'un repositori, l'aprovisionament de la màquina o la configuració de les carpetes compartides.

2.1.1. Linux Containers

Linux Containers és el nom del projecte darrere les tecnologies LXC, LXD i LXCFS que té com a objectiu proporcionar un entorn neutral per al desenvolupament de les tecnologies de contenidors a Linux.

- **LXC** és un conjunt d'eines per a la creació de contenidors sobre Linux.
- **LXD** proporciona una nova interfície per treballar amb LXC mitjançant una única eina de línia d'ordres i una forma de treballar més similar a la que s'utilitza habitualment amb màquines virtuals.
- **XLCFS** és un sistema de fitxer per noms d'usuari (FUSE, en anglès) que soluciona alguns problemes amb què es troben els usuaris que fan servir un sistema de contenidors.

Quan es treballa amb aquestes tecnologies es recomana utilitzar la distribució de Linux Ubuntu, ja que inclou totes les dependències necessàries, i Cannonical Ltd inclou suport a llarg termini (LTS o *long term support*, en anglès) per a LXC a les seves pròpies distribucions de tipus *LTS*.

El component LXC és la base del sistema de contenidors i el seu objectiu és crear un entorn el més proper possible a una instal·lació estàndard de Linux, però fent servir el mateix nucli del sistema operatiu de la màquina on s'executa.

Per altra banda, el component LXD fa servir la implementació de LXC internament, però afegeix un sistema de càrrega d'imatges per crear els contenidors. Els contenidors es gestionen de manera similar a com es faria si fossin màquines virtuals i permet realitzar aquestes operacions a través de la xarxa.

Un altre avantatge de fer servir LXD és que pot utilitzar-se conjuntament amb OpenStack (mitjançant el connector nova-lxd), de manera que es pot treballar al núvol tant amb contenidors com amb màquines virtuals de forma transparent de cara als usuaris. Això obre la porta a fer servir sistemes més avançats que incloguen servidors de càrrega i la creació automatitzada d'instàncies, per exemple, per augmentar el nombre de contenidors que serveixen una determinada aplicació segons el nombre d'usuaris connectats al sistema.

2.1.2. Contenidors: Docker

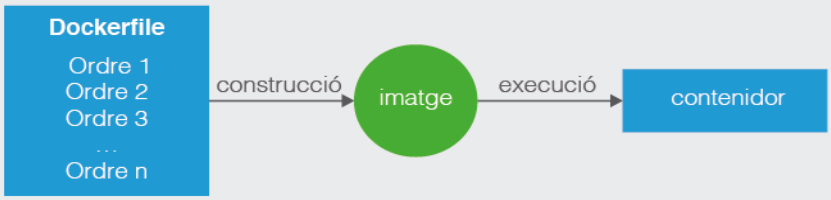
Docker és un sistema de contenidors basat en el nucli de Linux. Es tracta de programari lliure i entre els principals col·laboradors hi ha empreses com Google, IBM, Cisco, Microsoft i Red Hat.

Com que es tracta d'un contenidor basat en Linux, no pot utilitzar-se directament a màquines amb altres sistemes operatius, però és possible utilitzar-lo en altres entorns de desenvolupament mitjançant la virtualització. Al seu web hi ha enllaços a instal·ladors que simplifiquen aquesta tasca i inclouen tots els fitxers necessaris per instal·lar Docker a Linux, Mac i Windows.

Alguns avantatges de desplegar una aplicació utilitzant Docker són els següents:

- L'aplicació funciona igual en el servidor de proves que en el de producció perquè l'entorn és el mateix.
- Cada aplicació es troba aïllada de la resta d'aplicacions, en el seu propi contenidor.
- No cal instal·lar cap altre component, a banda de Docker, per executar l'aplicació en un altre equip. L'aplicació funciona directament en instal·lar el contenidor, perquè totes les dependències es troben al contenidor.

Els contenidors de Docker són instàncies d'una imatge que conté tots els fitxers necessaris empaquetats per crear el contenidor en un sol fitxer. Al seu torn, aquesta imatge és construïda a partir d'un fitxer anomenat Dockerfile, que conté totes les ordres necessàries per assemblar-la. Procés d'execució d'un contenidor:



El primer pas per treballar amb contenidors de Docker és instal·lar-lo per a utilitzar Docker amb Mac, Windows i diferents distribucions de Linux. La instal·lació és molt simple en tots els sistemes operatius, però en cas de dubte recordeu que podeu trobar tota la informació necessària a la pàgina de descàrrega.

Una vegada instal·lat, cal executar-lo. En la majoria de sistemes operatius, es necessiten permisos d'administrador per poder gestionar la xarxa. Per comprovar que s'ha instal·lat amb èxit, obriu una finestra amb la terminal o el símbol del sistema (segons quin sistema operatiu feu servir) i escriviu-hi *docker -v*.

El resultat ha de ser similar al següent:

```
[node1] (local) root@192.168.0.13 ~
$ docker -v
Docker version 24.0.7, build afdd53b
[node1] (local) root@192.168.0.13 ~
$
```

2.1.3. Creació i desplegament d'un contenidor amb Docker

Per veure el funcionament d'un contenidor, cal crear-ne un per executar una aplicació en PHP que mostre per pantalla el missatge: "Hola món!". Primer, creeu un directori anomenat *prova-docker*, on es desaran tots els fitxers d'aquest projecte. Dintre d'aquest directori, creeu-ne un altre anomenat *src* amb un fitxer de text pla anomenat index.php i el contingut següent:

```
<?php
echo "Hola món!";
```

Per poder executar aquesta aplicació és necessari un sistema operatiu, un servidor web (per exemple, Apache) i PHP. Per indicar a Docker que ha d'incloure la imatge, creeu un fitxer de text pla dins del directori *prova-docker* anomenat Dockerfile.

Repositori d'imatges per Docker

Podeu trobar imatges de Docker fiables per utilitzar com a base a l'enllaç següent: hub.docker.com. En aquest repositori hi ha tant imatges oficials (més fiables) com públiques.

Docker requereix el nom d'una imatge per fer-la servir com a base. Aquestes imatges inclouen els fitxers propis de la distribució (per exemple, Ubuntu o Debian) i en alguns casos algunes aplicacions preinstal·lades com PHP o MySql.

L'enllaç a la imatge de Docker oficial per a PHP es troba a l'enllaç següent: hub.docker.com/_/php/. Al principi de la pàgina hi ha la llista d'imatges disponibles.

Fixeu-vos que a cada fila es mostren, d'esquerra a dreta, les opcions disponibles, de la més concreta a la més genèrica. És a dir, si s'especifica 7.1.3-apache es farà servir la imatge amb la versió 7.1.3 de PHP i Apache, mentre que si s'especifica en el fitxer 7-apache es farà servir una versió de PHP 7, però no sabreu quina (habitualment la més recent que s'haja afegit a aquest repositori). En un cas encara més extrem, si només s'especifica apache, la versió de PHP podria ser qualsevol (per exemple, PHP 8 o 9).

Per aquests motius es recomana fer servir sempre una versió concreta. En cas contrari, es poden produir incompatibilitats en les aplicacions i, fins i tot, pot passar que a partir d'una mateixa imatge base es generen diferents imatges, ja que la versió pot canviar en qualsevol moment.

Habitualment els números de versió d'un programa indiquen les diferències següents:

- El primer indica el número de la versió, i no acostuma a ser completament compatible amb l'anterior (per exemple, PHP 7.0 no és compatible amb PHP 5.6).
- El segon número s'utilitza quan s'afegeixen noves funcionalitats.
- El tercer número indica correccions.

Així doncs, a l'hora de seleccionar una imatge, normalment només cal indicar els dos primers números de versió. Per exemple, si s'indica 7.1 com a versió, s'inclou la versió més recent amb totes les correccions actualitzades.

Per indicar a Docker la imatge base, es fa amb l'ordre FROM seguida del nom del repositori (php), dos punts (:) i el nom de la imatge (per exemple, 7.1-apache):

FROM php:7.1-apache

Com que es vol copiar l'aplicació dins del contenidor, cal utilitzar l'ordre COPY indicant la ruta d'origen i la ruta de destí:

COPY src/ /var/www/html

Aquesta és la ruta que utilitza aquesta imatge en concret, que està basada en la distribució de Debian de Linux. Per saber a quina distribució pertany una imatge i tot el que conté, només cal clicar l'enllaç a la dreta de cada fila per accedir al fitxer Dockfile utilitzat per crear-la.

Finalment, cal indicar a Docker que cal exposar el port 80 del contenidor per poder accedir a la pàgina web. Per fer-ho s'utilitza l'ordre EXPOSE:

EXPOSE 80

De tot açò tenim que el contingut del fitxer Dockerfile per a crear la imatge ha de ser:

```
FROM php:7.1-apache
COPY src/ /var/www/html
EXPOSE 80
```

Una vegada creat el fitxer Dockerfile i desat dins de la carpeta *prova-docker*, per generar la imatge heu d'escriure a la línia d'ordres:

docker build -t img1 .

El paràmetre -t es fa servir per indicar el nom de la imatge (en aquest cas, "img1") i el punt final indica que es crearà en el mateix directori. Una vegada es prema la tecla retorn, començaran a descarregar-se els fitxers necessaris per crear la imatge.

El resultat ha de ser similar al següent:

```
xavi@xavi:~/prova-docker$ sudo docker build -t img1 .
Sending build context to Docker daemon  3.584kB
Step 1/3 : FROM php:7.1-apache
7.1-apache: Pulling from library/php
000eee12ec04: Pull complete
8ae4f9fcfeea: Pull complete
60f22fbbd07a: Pull complete
ccc7a63ad75f: Pull complete
a2427b8dd6e7: Pull complete
91cac3b30184: Pull complete
d6e40015fc10: Pull complete
54695fdb10a7: Pull complete
500ca11be45f: Pull complete
86b2805859cf: Pull complete
c61685fa4f4f: Pull complete
0bf989f9dbbb: Pull complete
01848ea209b5: Pull complete
Digest: sha256:70eefcf4372b279101259e905996b7733a87688d0b48625af01b55947836bb1f
Status: Downloaded newer image for php:7.1-apache
 ---> b9858ffdd4d2
Step 2/3 : COPY src/ /var/www/html
 ---> ab8ce03fc168
Step 3/3 : EXPOSE 80
 ---> Running in fdc26c611b15
Removing intermediate container fdc26c611b15
 ---> 615b380f5b63
Successfully built 615b380f5b63
Successfully tagged img1:latest
```

Seguidament, per executar el contenidor, heu d'escriure des de la línia d'ordres:

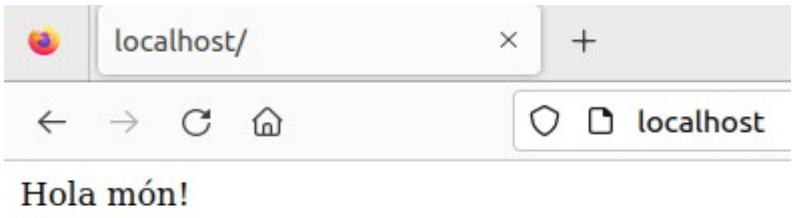
```
sudo docker run -p 80:80 img1
```

L'opció run indica que es vol executar una instància de la imatge "img1". Fixeu-vos que el nom de la imatge ha d'anar al final de l'ordre. L'opció -p indica la redirecció de ports (és el primer el port de la màquina hoste i el segon el port del contenidor). És a dir, s'executarà el contenidor "hola-mon" i es podrà accedir al seu port 80 des del port 80 de la màquina hoste.

El resultat d'executar-lo ha de ser similar al següent:

```
xavi@xavi:~/prova-docker$ sudo docker run -p 80:80 img1
AH00558: apache2: Could not reliably determine the server's fully qualified domai
n name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress thi
s message
AH00558: apache2: Could not reliably determine the server's fully qualified domai
n name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress thi
s message
[Tue Nov 07 16:28:11.179995 2023] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.
4.38 (Debian) PHP/7.1.33 configured -- resuming normal operations
[Tue Nov 07 16:28:11.180075 2023] [core:notice] [pid 1] AH00094: Command line: 'a
pache2 -D FOREGROUND'
```

Tot i que es mostren missatges d'advertència d'Apache, l'aplicació ha de funcionar correctament. Per comprovar-ho només heu d'obrir el vostre navegador i introduir com a URL "localhost". Hauria de mostrar-se per pantalla el missatge "Hola món!".



En cas de voler utilitzar un port diferent de la màquina hoste (per exemple, el 8080, per accedir des de l'URL localhost:8080), només cal executar la imatge canviant aquest port, tal com es mostra en l'exemple següent:

```
sudo docker run -p 8080:80 img1
```

Si proveu de modificar el fitxer index.php i actualitzeu la pàgina al navegador, veureu que els canvis no s'hi veuen reflectits. Això és d'esperar, perquè l'aplicació s'ha copiat dins del contenidor. Si es vol actualitzar l'aplicació, cal tornar a construir el contenidor.

Hi ha casos en què aquest comportament no és el desitjat (per exemple, durant el desenvolupament). Per solucionar aquest problema, Docker ofereix l'opció de muntar volums que funcionaran com a directoris compartits entre la màquina hoste i el contenidor.

Per muntar un volum, s'ha de fer des de la línia d'ordres utilitzant l'opció -v i indicant el nom del directori de la màquina hoste, dos punts (:), i el nom del directori al contenidor.

```
sudo docker run -p 80:80 -v /provar-docker/src/:/var/www/html/ hola-mon
```

Cal destacar que s'ha d'utilitzar la ruta absoluta. No és vàlid fer servir ~ per indicar que es tracta de la carpeta de l'usuari actual a Linux o Unix, ni .. per indicar que es tracta del directori pare.

Fixeu-vos que, per muntar un contenidor, s'ha de fer des de la línia d'ordres i no al fitxer Dockerfile: d'aquesta manera s'evita que es trenque la portabilitat. Com que es tracta de recursos externs al contenidor, Docker no pot assegurar que aquests estiguen disponibles en qualsevol equip.

Si llisteu el contingut del vostre directori, no veureu la imatge creada enlloc. Això és normal: no forma part de la vostra aplicació i no tindria sentit que es mostrés en aquest directori. Per veure un llistat de les imatges instal·lades, s'utilitza l'ordre docker images i el resultat serà similar al següent:

```
xavi@xavi:~/prova-docker$ sudo docker images
REPOSITORY                  TAG         IMAGE ID       CREATED          SIZE
img1                        latest      615b380f5b63   13 minutes ago   401MB
myimage1                    1.0         c775a8132947   10 hours ago     123MB
```

Per poder desplegar l'aplicació l'heu de pujar al repositori d'imatges de Docker. Per fer-ho, necessiteu crear un compte a Docker Hub (hub.docker.com). Una vegada creat el compte i confirmat mitjançant l'enllaç rebut al correu, podeu connectar amb la pàgina.

Per poder pujar la imatge al repositori, heu d'autenticar-vos amb el nom d'usuari i la contrasenya des de la línia d'ordres, fent servir l'ordre docker login. No cal passar cap paràmetre, però us demanarà el nom d'usuari i la contrasenya. El resultat serà similar al següent:

```
xavi@xavi:~/prova-docker$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't h
ave a Docker ID, head over to https://hub.docker.com to create one.
Username: xaviblanes
Password:
WARNING! Your password will be stored unencrypted in /root/snap/docker/2904/.dock
er/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

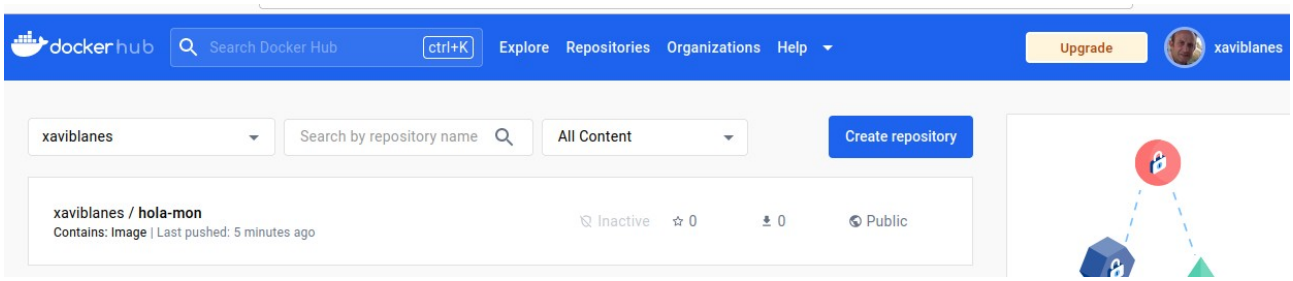
```
xavi@xavi:~/prova-docker$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS     NAMES
f4ffda90072a   img1      "docker-php-entrypoi…"   30 minutes ago   Exited (0) 20 minutes ago             great_banach
```

```
xavi@xavi:~/prova-docker$ sudo docker commit great_banach xaviblanes/hola-mon
sha256:63941bd1e34232dd29a086721d544710114a79016410859b098431e41cfdcebd
```

```
xavi@xavi:~/prova-docker$ sudo docker push xaviblanes/hola-mon
Using default tag: latest
The push refers to repository [docker.io/xaviblanes/hola-mon]
00a2728e9ed1: Pushed
3adcb233aee1: Pushed
0817436a8f49: Mounted from library/php
3385a426f542: Mounted from library/php
35c986c7de74: Mounted from library/php
53bab0663330: Mounted from library/php
606c36b65880: Mounted from library/php
ab99fcc1a184: Mounted from library/php
9691e5d7a4c7: Mounted from library/php
6a4d393f0795: Mounted from library/php
e38834ac7561: Mounted from library/php
ec64f555d498: Mounted from library/php
840f3f414cf6: Mounted from library/php
17fce12edef0: Mounted from library/php
831c5620387f: Mounted from library/php
latest: digest: sha256:c0dafabacfc8ab676f04c8618a96c26aef82a9dc
xavi@xavi:~/prova-docker$
```

Llistem els contenidors llançats, copien el alias (great_banach), fen un commit i finalment un push.

Ja està pujat:



Altres aspectes de la utilització de Docker que cal tenir en compte són els següents:

- Quan acaba la tasca que s'està executant al contenidor, el contenidor es para automàticament.
- Cada aplicació s'executa en un contenidor diferent, ja que cada contenidor està lligat a un únic procés.

- Un mateix equip pot tenir múltiples contenidors funcionant al mateix temps. Per veure els contenidors en execució es pot fer servir l'ordre docker ps -a2.
- Docker fa servir un sistema de capes per generar les imatges; aquestes capes es corresponen amb les línies del fitxer Dockerfile.
- Per automatitzar el desplegament s'acostuma a utilitzar altres eines i serveis de desplegament com Kubernetes (kubernetes.io) o Ansible (www.ansible.com).

2.1.4. Virtualització. Vagrant

Vagrant (www.vagrantup.com) és una tecnologia basada en la virtualització que permet crear entorns de desenvolupament portables i fàcilment reproduïbles. Gràcies a la virtualització és possible treballar amb diferents versions de sistemes operatius a la mateixa màquina.

Cal tenir en compte que a l'hora de treballar en diferents projectes, les característiques de les màquines de producció molt rarament seran les mateixes que les de l'equip propi de desenvolupament. És a dir, ni el sistema operatiu, ni les versions dels servidors de bases de dades, ni les versions dels servidors web es correspondran.

Per entendre quin és el problema que soluciona Vagrant vegeu l'exemple següent:

Treballar en múltiples projectes localment

Un professional treballa en dos projectes al mateix temps com a desenvolupador, i la màquina de desenvolupament utilitza Windows 10.

- El projecte actual és una aplicació en PHP 7, desplegada en un servidor Red Hat Enterprise Linux 7.3, que fa servir NGINX com a servidor web.
- S'ha detectat un problema en un projecte antic que s'ha de solucionar. Aquest està desenvolupat en PHP 5.3, desplegat en un servidor amb Ubuntu 12.02, que fa servir Apache com a servidor web.

Per poder treballar amb tots dos projectes alhora en la mateixa màquina cal fer les accions següents:

- Instal·lar dos servidors web diferents en la vostra màquina.
- Instal·lar dues versions diferents de PHP.
- Modificar les configuracions per evitar conflictes. (A producció no s'ha de fer; si aquest canvi es propaga als servidors, pot provocar errors.)

S'ha de tenir en compte que cadascun d'aquests serveis consumeix recursos, s'utilitzen o no, a la màquina principal.

Cal destacar un altre inconvenient: no es poden descobrir errors deguts al sistema operatiu fins que no es faça el desplegament, ja que el de la màquina no es correspon amb els dels projectes.

Com es pot apreciar a l'exemple, en el millor dels casos s'han hagut d'instal·lar múltiples serveis a l'ordinador i no hi ha cap garantia que quan es desplegue, l'aplicació funcione correctament. A més, si a l'equip hi ha més d'un desenvolupador treballant en els

mateixos projectes, s'han de repetir aquests passos per a cadascun dels entorns de cada desenvolupador implicat.

Penseu que les configuracions d'aquests altres equips també poden ser diferents unes de les altres, i possiblement no han treballat en els mateixos projectes. Per conseqüent, quan s'instal·len els nous serveis, es poden presentar nous problemes i conflictes.

Una altra opció és treballar directament sobre servidors remots. En aquest cas, per a cada projecte s'aconsegueix una configuració molt aproximada a la de desenvolupament, però presenta els inconvenients següents:

- S'incrementa el cost de desenvolupament, ja que s'han de pagar servidors extres per a cada projecte.
- Múltiples desenvolupadors no poden treballar sobre el mateix servidor directament, ja que s'han de transferir els canvis i es podrien sobreescriure els fitxers.
- Segons la configuració d'aquests servidors i altres eines de desenvolupament, moltes tasques s'han de realitzar utilitzant la línia d'ordres i editors de text que no admeten l'ús del ratolí.
- En cas que hi hagués problemes de connectivitat, no es podria continuar treballant.

En canvi, si s'utilitzen màquines virtuals, es pot treballar amb tantes configuracions diferents com siga necessari, independentment del sistema operatiu original. Aquest sistema soluciona pràcticament tots els problemes detectats als mètodes anteriors:

- Les configuracions estan encapsulades, no hi ha conflictes entre l'una i l'altra.
- La configuració és molt aproximada a la de la màquina de desenvolupament.
- Cada desenvolupador treballa a la seua màquina, no hi ha perill de perdre dades.
- Es poden utilitzar les eines de desenvolupament preferides pel programador, ja que es poden passar les dades entre màquines utilitzant carpetes compartides.
- No hi ha cap cost afegit perquè no s'han de contractar nous servidors.
- No es poden produir problemes de connectivitat, ja que tot es troba a la mateixa màquina.

Tot i que és una molt bona opció, presenta l'inconvenient d'haver de crear múltiples màquines virtuals quan es treballa en un equip de desenvolupadors, ja que a cada ordinador s'ha de crear la màquina virtual per al projecte i fer tot el procés d'instal·lació del sistema operatiu, els serveis i la configuració o copiar els fitxers d'una de les màquines de l'altre desenvolupador al nou.

Afortunadament, Vagrant soluciona aquest problema fent servir un fitxer de configuració que permet, ràpidament, recrear qualsevol màquina sense interacció de l'usuari, tot i que una vegada estan instal·lades, els desenvolupadors hi poden interactuar utilitzant la línia d'ordres o directament, mitjançant una carpeta compartida.

A més, quan no es necessiten, es poden destruir per recuperar l'espai al disc, ja que es necessita molt poc de temps per recrear-les i no és gens complex. En aquest cas, només es perden les dades creades dins de la màquina: les de la carpeta compartida no es perden, que és la carpeta on es recomana treballar.

Un altre avantatge és que Vagrant permet generar múltiples màquines a partir d'un sol fitxer de configuració, de manera que es poden simular interaccions complexes que no són possibles quan es treballa amb un únic ordinador.

En resum, la utilització de Vagrant presenta els avantatges següents:

- L'entorn de desenvolupament és similar al de producció.
- Es poden mantenir múltiples entorns de desenvolupament a la mateixa màquina.
- Es pot desplegar l'entorn de desenvolupament en qüestió de minuts.
- Tots els membres de l'equip de desenvolupament treballen amb el mateix entorn.
- Es treballa amb el codi mitjançant carpetes compartides.
- Quan un projecte no és necessari, es pot destruir la màquina virtual, perquè per reconstruir-la (si calgués en el futur) només es necessita el fitxer Vagranfile.
- Es poden simular conjunts de màquines.

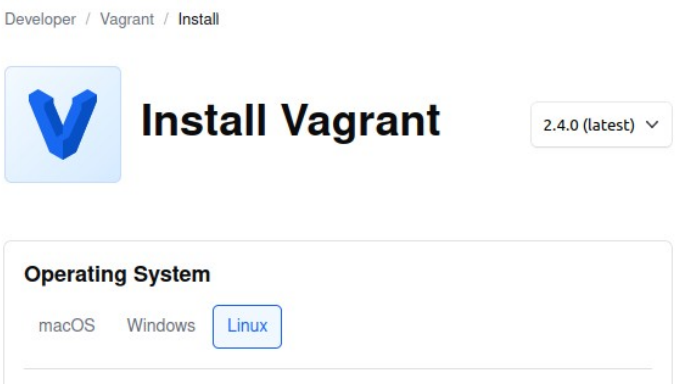
2.1.5. Instal·lació d'una caixa amb Vagrant

A Vagrant, *proveïdor* fa referència a un programari de virtualització.

Per instal·lar una màquina virtual utilitzant Vagrant, primer cal instal·lar el programari indispensable. Independentment del sistema operatiu, cal instal·lar Vagrant i un proveïdor (*provider*) per poder crear les caixes (*boxes*), que és el nom donat a les màquines virtuals empaquetades.

El proveïdor recomanat és VirtualBox, perquè és gratuït i està disponible a totes les plataformes que són admeses per Vagrant: macOS, Linux i Windows.

De la web de Vagrant: *https://www.vagrantup.com/*



O bé directament: `sudo apt install vagrant`

I per a VirtualBox: `sudo apt install virtualbox`

Una vegada s'ha instal·lat Vagrant i VirtualBox i es disposa d'un client SSH operatiu, es pot procedir a instal·lar la primera caixa de Vagrant. Per fer-ho, cal indicar el nom de la caixa. Aquest ha de ser un dels disponibles a app.vagrantup.com/boxes/search (que ja estan inclosos a la configuració de Vagrant) o un de nou que afegiu vosaltres.

Anem a crear una VM de CentOS 8 amb Vagrant. Primer crearem una carpeta:

```
$ mkdir ~/projecte-vagrant
```

```
$ cd projecte-vagrant
$ vagrant init centos/8
```

i ara:

```
$ vagrant up
$ vagrant ssh
```

i ja estem dins de MV de Centos 8. Si volem eixir fem

```
exit.
```

Per a parar la màquina

```
$ vagrant halt
```

i per a destruir tots els recursos creats

```
$ vagrant destroy
```

Opcions de configuració

Es poden trobar totes les opcions de Vagrant a l'enllaç següent:

www.vagrantup.com/docs/.

Vagrant disposa d'una gran quantitat d'opcions de configuració, però les que s'han de modificar més habitualment són les següents:

- Configuració de carpetes compartides
- Configuració de la xarxa
- Aprovisionament

La configuració de les carpetes compartides es troba al fitxer de configuració a la variable config.vm.synced_folder. Descomentant aquesta variable es pot canviar la configuració de les carpetes. S'ha d'especificar, com a primer paràmetre, la carpeta de la màquina real i, com a segon paràmetre, la carpeta que correspondrà a la caixa.

Quant a la configuració de la xarxa, es pot diferenciar entre tres modes:

- ***Port forwarding***: no permet utilitzar ports per sota de 1024. Utilitza els ports de la màquina real redirigint les peticions a la caixa. Per exemple, el port 2222 de la màquina real enllaça amb el port 22 de la virtual en la configuració per defecte. Un altre desavantatge és que no permet utilitzar el protocol UDP i no pot connectar amb altres màquines virtuals.
- ***Host-only***: no es pot accedir a la caixa des d'altres equips de la xarxa, només s'hi pot accedir des de la màquina on s'ha iniciat i des d'altres caixes al mateix ordinador.
- ***Bridged networking***: la caixa es connecta a l'encaminador com un altre equip, disposa de la seva pròpia adreça IP a la xarxa i s'hi pot accedir des de qualsevol equip.

Aprovisionar (*provisioning*) és instal·lar programari i executar accions addicionals automàticament en crear una caixa.

Habitualment, les caixes de tercers es fan servir com a base i, seguidament, s'aprovisionen amb els serveis extres concrets que es necessitin. Per exemple, es pot crear una caixa amb Ubuntu 12.02 aprovisionada amb Apache, PHP i MySQL, per fer servir com a servidor web.

Entre els sistemes d'aprovisionament admesos per Vagrant hi ha: Puppet, Chef, Ansible, Salt i Docker.

Per aprovisionar una màquina es poden fer servir diferents sistemes d'aprovisionament: el més bàsic és *shell*, que permet utilitzar les ordres de Linux per portar a terme l'aprovisionament. Per exemple, per fer la instal·lació automàtica del servidor web Apache2 es pot afegir (o descomentar) al fitxer Vagrantfile.

```
config.vm.provision "shell", inline: <<-SHELL
  apt-get update
  apt-get install -y apache2
SHELL
```

S'ha fet servir el paràmetre -y en instal·lar Apache2. Això indica que s'ha de seleccionar *yes* a totes les preguntes que es fan durant la instal·lació. Si la màquina ja ha estat aprovisionada anteriorment (per exemple, si no és la primera vegada que s'inicia), cal utilitzar l'ordre vagrant provision.

Un clúster és un conjunt de màquines interconnectades mitjançant una xarxa per treballar com una sola unitat.

Una altra de les opcions disponibles en treballar amb Vagrant és la possibilitat de generar múltiples caixes per formar un clúster multimàquina que siguin generades a partir d'un mateix fitxer. D'aquesta manera, poden inicialitzar-se dues o més màquines amb els seus propis aprovisionaments per fer que treballin juntes. Per exemple, es pot afegir una caixa com a servidor web i una altra com a servidor de bases de dades afegint les línies següents al fitxer "Vagrantfile":

```
Vagrant::Config.run do |config|
  config.vm.box = "ubuntu/trusty64"

  config.vm.define "web" do |web|
    web.vm.forwarded_port 80, 8080
    web.vm.provision :shell, path: "web-provisio.sh"
    web.vm.network :hostonly, "192.168.33.10"
  end

  config.vm.define "bd" do |bd|
    bd.vm.provision :shell, path :"bd-provisio.sh"
    bd.vm.network :hostonly, "192.168.33.11"
  end
end
```

Les màquines creades com a clúster, una vegada inicialitzades, es poden iniciar, aturar i fins i tot destruir individualment.

En aquest cas, s'ha fet servir un fitxer amb les comandes que cal utilitzar per portar a terme l'aprovisionament. Aquests fitxers són guions de Linux amb les ordres que s'ha de portar a terme a cada màquina.

Aquest fitxer crea una màquina caixa, anomenada “bd”, que redirigeix l’entrada del port 8080 de l’ordinador físic al port 80 de la màquina virtual i l’aprovisiona amb el contingut del fitxer “web-provisio.sh” (que podria contenir la instal·lació d’Apache, del client MySQL i PHP) i estableix el mode de xarxa “hostonly” i la IP 192.168.33.10.

La segona màquina, anomenada “bd”, és aprovisionada amb el contingut del fitxer bd-provision.sh (que podria contenir la configuració del servidor MySQL) i estableix el mode de xarxa *host-only* amb IP 192.168.33.11.

Com que se sap l’adreça de totes dues caixes, és possible accedir al servidor MySQL des de la caixa amb el servidor web. S’ha de tenir en compte que per poder connectar aquestes màquines entre si, la configuració de xarxa ha de ser *host-only* (no pot accedir-s’hi des d’altres equips) o *bridge networking* (altres equips podran connectar-s’hi).

Resum d'ordres de Vagrant

Aquí es presenta un resum de les ordres més utilitzades per treballar amb Vagrant:

- vagrant -v: mostra la versió de Vagrant instal·lada.
- vagrant init nom: inicialitza la màquina amb el nom indicat.
- vagrant up: posa en marxa la caixa corresponent a la carpeta actual, ja estiga parada o en suspensió, i si és necessària la descàrrega.
- vagrant box -add url: afegeix la caixa que es troba a l’URL a la llista de caixes disponibles.
- vagrant box -list: llista les caixes instal·lades (aquestes són globals, poden ser utilitzades per a múltiples projectes).
- vagrant ssh: connecta a la caixa mitjançant SSH.
- vagrant suspend: posa la caixa en suspensió.
- vagrant resume: reactiva una caixa en suspensió.
- vagrant halt: atura la màquina virtual.
- vagrant detroy: elimina la caixa del disc, però no de la llista de caixes.
- vagrant status: mostra l’estat de les caixes inicialitzades.
- vagrant reload: reinicia la caixa i recarrega el fitxer Vagrantfile.
- vagrant provision: reaprovisiona la caixa.
- vagrant package: permet empaquetar una caixa activa.

Cal destacar l’ordre vagrant package, que permet reempaquetar una caixa amb tots els canvis que s’hagen realitzat internament amb un altre nom. Per exemple, si a la caixa heu instal·lat Apache, MySQL i PHP, la podeu reempaquetar per distribuir-la:

vagrant package --output el-meu-entorn-lamp.box

Es recomana fer servir aquest mètode en lloc d’utilitzar l’aprovisionament quan les accions que s’han de realitzar són complexes o s’executen lentament. Per exemple, en el cas d’haver d’instal·lar múltiples serveis o haver de realitzar instal·lacions on es requereixi la interacció de l’usuari.

2.2. Servei de sistema de noms de domini (DNS)

El sistema de noms de domini (*domain name system* o DNS, en anglès) és un servei encarregat de convertir els noms de domini (com www.google.com) en l'adreça IP corresponent (per exemple, 142.250.184.4), comparable a una guia telefònica adaptada a internet, però amb la diferència que és possible actualitzar-la automàticament sense afectar els usuaris.

Un nombre de 8 bits està en el rang que va de 0 fins a 255, mentre que un nombre de 16 bits està en el tram entre 0 i 65.635.

Cal tenir en compte que cada equip connectat a internet té una adreça IP que l'identifica. Aquesta adreça pot estar formada per 4 nombres de 8 bits (protocol IPv4) o 8 nombres de 16 bits (protocol IPv6), però no totes les adreces corresponen a un domini concret.

Per altra banda, cal distingir entre les adreces IP públiques i privades. Les adreces públiques són accessibles des d'internet, mentre que les privades estan reservades i són utilitzades per configurar les xarxes locals.

A més, no totes les adreces públiques corresponen a un domini, ja que potser no estan assignades o estan assignades a proveïdors de serveis d'internet. Per exemple, els proveïdors tenen assignats un rang d'adreces que són utilitzades pels seus clients. Per aquesta raó, a cada usuari que es connecta a internet se li assigna una IP pública. Aquesta IP pública és utilitzada pel punt d'accés (per exemple, un encaminador), i habitualment també té una IP privada que correspon a la seva adreça IP dins de la xarxa local (assignada per l'administrador del sistema o l'encaminador).

També cal distingir entre un servidor de DNS local, encarregat de gestionar els noms de domini dintre d'una xarxa privada que facilita identificar els equips d'una empresa, i els servidors DNS d'internet, que són els que permeten identificar els servidors que allotgen les pàgines web i altres serveis.

Cada proveïdor d'internet acostuma a configurar als seus encaminadors els seus propis servidors de DNS, però cada usuari pot canviar-ne la configuració per utilitzar el que prefereixi (per exemple, el servidor DNS de Google).

Tot i que actualment no és habitual trobar-se problemes amb els servidors de DNS globals, si un o més dels servidors DNS configurats han caigut, és possible que algunes adreces deixin de funcionar, perquè no es pot realitzar la traducció de nom de domini a adreça IP.

Un altre problema molt més habitual és que en registrar un nom de domini i assignar-lo a una adreça IP, o canviar l'adreça IP a la qual apunta un nom de domini, aquest pot trigar fins a 24 hores a propagar-se a tots els servidors DNS. És a dir, hi ha usuaris que poden accedir correctament a la nova pàgina, mentre que d'altres continuen sense poder-hi accedir o accedint a l'adreça anterior.

Tingueu en compte que segons la legislació del país en el qual us trobeu (o les polítiques dels mateixos proveïdors) és possible que els proveïdors de serveis d'internet bloquegin l'accés a determinats dominis. Això ho fan a través dels seus serveis de DNS, de manera

que es poden evitar aquestes restriccions canviant la configuració del mòdem o encaminador per utilitzar algun altre servidor de DNS.

Cal tenir en compte que per resoldre un nom de domini s'ha de realitzar una connexió amb el servidor remot i esperar a rebre la resposta amb l'adreça IP abans d'accedir-hi. Així doncs, és possible que alguns servidors de DNS siguin més ràpids que altres, ja siga per la potència del servidor de DNS o per la localització respecte a l'usuari que realitza la petició.

2.2.1. Com funciona un servei DNS

El sistema de noms de domini és jeràrquic, és a dir, consisteix en una estructura de dades amb forma d'arbre subdividit en zones, que poden consistir en un o múltiples dominis i subdominis. Cada node i fulla d'aquest arbre consta d'una etiqueta i pot contenir cap o múltiples registres de recursos, a excepció de l'arrel, que té com a etiqueta una cadena buida.

La jerarquia de dominis descendeix de dreta a esquerra, de manera que el domini de primer nivell (*top-level domain o TLD*, en anglès) són les lletres que es troben més a la dreta, precedides per un punt. Per exemple, per al domini ioc.xtec.cat, el domini de primer nivell és .cat.

Fixeu-vos que els noms de dominis estan dividits en fragments separats per punts. Cadascun d'aquests fragments es correspon amb una etiqueta d'un node o fulla de l'arbre.

Cada etiqueta cap a l'esquerra es considera un subdomini de l'etiqueta a la seua dreta. Per tant, l'adreça ioc.xtec.cat es pot interpretar de la manera següent:

- cat és el domini de primer nivell.
- xtec és un subdomini de cat.
- ioc és un subdomini de xtec.

El sistema de noms de dominis està mantés per un sistema de bases de dades distribuïdes, en què els nodes d'aquesta base de dades són els servidors de noms (*name servers*, en anglès). Cada domini disposa, com a mínim, d'un servidor DNS que s'encarrega de publicar la informació sobre el domini a altres servidors per dalt seu a la jerarquia fins a arribar als servidors de noms arrel.

Normalment les consultes als servidors DNS són redirigides a servidors pròxims a l'usuari per a que augmente la velocitat de la resposta.

Quan es realitza una consulta al servidor DNS (suposant que no fa servir un sistema de cau), el servidor no resol el nom, sinó que retorna l'adreça d'un altre servidor més específic al qual cercar. Per exemple, si es demana l'adreça corresponent a ioc.xtec.cat retorna l'adreça del servidor DNS que gestiona els dominis de primer nivell cat. Al seu torn, aquest retorna l'adreça corresponent al subdomini xtec, que retorna l'adreça del següent servidor de la jerarquia, i el procés continua fins que s'obté la resposta d'un servidor autoritari (*authoritary server*, en anglès).

A la pràctica, gràcies als sistemes de cau, les consultes al servidor de noms arrel són quasi inexistents.

Per millorar el rendiment dels servidors de DNS, s'utilitza un sistema de cau que emmagatzema les adreces consultades prèviament. D'aquesta manera, el servidor pot resoldre la consulta automàticament sense haver de consultar els servidors de noms arrel i fer tot el recorregut cada vegada que es demana una adreça.

2.2.2. Configuració dels servidors DNS i els fitxers de zona

Instal·lació de BIND a Ubuntu

El programari de DNS més utilitzat a internet és BIND (Berkeley Internet Name Domain) i és considerat l'estàndard *de facto* en els sistemes operatius bastats en UNIX (com Linux). La primera versió d'aquest programari va ser creada a principis dels anys 80 i emmagatzemava les dades mitjançant fitxers de text pla però, a partir de l'any 2007, amb la versió 9.4 es va incloure l'opció d'utilitzar una gran varietat de sistemes d'emmagatzematge, incloent-hi LDA, PostgreSQL i MySQL.

Directori actiu és el nom de la implementació del servei de directori de Microsoft per al sistema operatiu Windows.

En el cas de les xarxes privades, és possible utilitzar un servidor DNS que s'encarregue de gestionar les conversions dels noms dels recursos de la xarxa (ordinadors, impressores i altres dispositius) a les seves adreces IP. En aquest cas, com que la major part dels equips acostumen a tenir instal·lat Microsoft Windows i en molts casos depenen de la tecnologia de directori actiu (*active directory*, en anglès), s'acostuma a utilitzar el servidor de DNS de Windows.

Mentre que BIND es configura mitjançant la línia d'ordres i editant fitxers de text pla, el servidor DNS de Windows es configura utilitzant la interfície gràfica, fet que pot simplificar la tasca als usuaris novells.

A l'hora de configurar un servei de DNS, cal distingir entre els diferents tipus de servidors:

- **Servidor de noms cau** (conegut en anglès com a *resolver*): servidor responsable d'emmagatzemar les adreces resoltes per accelerar futures consultes.
- **Servidor mestre**: servidor principal, responsable de gestionar els fitxers de zona i notificar als servidors esclaus quan s'hi han produït canvis.
- **Servidor esclau**: servidor que, periòdicament, es connecta al servidor mestre per rebre actualitzacions.

Tingueu en compte que, depenent de la configuració del servei, un mateix servidor pot encarregar-se d'una o més funcions. Així, podem trobar que un servidor és mestre d'altres servidors i esclau d'un altre, simultàniament.

Fitxers de zona

Els fitxers de zona formen la part principal d'un servidor DNS i contenen tota la informació necessària per poder resoldre les consultes.

Vegeu a continuació un exemple de fitxer de zona de tipus SOA:

```
@ IN  SOA example.com. root.example.com. (
                              2 ; Serial
```

```
                604800 ; Refresc
                 86400 ; Reintentar
               2419200 ; Expiració
                604800 ) ; Cau negatiu (Temps de vida)
```

A l'exemple anterior el significat de cada fragment és el següent:

- **@**: és una drecera pel valor de la directriu $ORIGIN, l'origen actual.
- **IN**: indica que es tracta d'una adreça d'internet.
- **SOA**: és el tipus de registre de recursos (*resource record*, en anglès).
- **exemple.com**: correspon al domini del servidor primari de dades
- **root.example.com.**: és l'adreça de l'administrador del servidor DNS. Cal destacar que l'adreça real seria *root@example.com*.

Els significats dels paràmetres entre parèntesis són els següents (cal tenir en compte que l'ordre és important):

- **Serial**: aquest nombre s'ha de modificar cada vegada que es fan canvis en el fitxer de zona. Serveix per indicar a altres servidors que s'han produït canvis, ja que compararan aquest serial amb el nombre de l'última consulta al servidor.
- **Refresc**: indica al servidor esclau cada quants segons ha de connectar-se per actualitzar la informació.
- **Reintentar**: en cas que es produeixi un error en la comunicació a l'hora de fer un refresc, indica quant de temps ha d'esperar per tornar a intentar-ho.
- **Expiració**: si el servidor esclau no pot connectar amb el servidor mestre, un cop es sobrepassa aquesta quantitat de temps, es considera que el fitxer de zona és invàlid.
- **Cau negatiu**: indica a partir de quant de temps deixa de ser vàlida la informació del cau.

Els tipus més importants de registres de recursos són els següents:

- **SOA** (*start of authority*): defineix els paràmetres de la zona (el domini). Només pot haver-n'hi un per fitxer de zona i ha de ser el primer registre de recursos.
- **NS** (*name server*): indica a quin servidor es pot trobar la resposta a la consulta.
- **A** (*adress*): especifica la traducció d'un nom de domini a l'adreça IP corresponent.
- **CNAME** (*canonical name*): s'utilitzen com a àlies d'una adreça definida per a un registre de recursos de tipus A.
- **PTR** (*pointer*): realitza la funció inversa als registres A. S'encarrega de fer la traducció d'adreces IP a noms de domini i s'utilitza únicament en els fitxers de zona inversos.
- **MX** (*mail exchanger*): s'utilitza per resoldre adreces de correu electrònic.

Configuració dels registres de recursos a un proveïdor de dominis

Per poder utilitzar un nom de domini a internet (per exemple, per a una pàgina web) s'ha de contractar un proveïdor, com pot ser Namecheap (www.namecheap.com) o Fundació.cat (www.fundacio.cat).

Una vegada s'ha contractat el domini, s'han de crear els registres per indicar a quina adreça IP s'han de dirigir les peticions que consulten aquest nom. Així doncs, si el vostre domini s'anomena domini.com i l'adreça IP pública del servidor on heu desplegat l'aplicació és 77.246.38.64 necessitareu afegir un registre de recursos de tipus A al fitxer de zona.

El més habitual és que no tingueu accés al servidor de DNS, però pràcticament tots els proveïdors de dominis proporcionen una interfície per poder afegir i eliminar registres de recursos de tipus A, CNAME i MX. En cas contrari, heu de contactar directament amb el proveïdor i proporcionar-li la informació necessària perquè agregue els registres.

A la taula següent podeu veure un exemple de configuració dels registres per a un domini en un proveïdor de dominis. Com es pot veure a la taula, s'han configurat dos registres de tipus A, un per al domini i un altre per a un subdomini, que apunten a adreces IP diferents (és a dir, es tracta de servidors diferents). Es pot veure que s'han creat 3 àlies (tipus CNAME): dos per redirigir les peticions dels subdominis www i ftp al domini principal, i el tercer per apuntar al servidor de correu sortint (aquesta correspondria al protocol POP3). Finalment, hi ha un registre de tipus MX utilitzat per indicar el servidor de correu entrant (que correspondria al protocol SMTP).

Taula: Exemple de configuració dels registres en un proveïdor de dominis.

Nom	**Tipus**	**Valor**
domini.com	A	77.246.38.64
sub.domini.com	A	89.248.102.203
ftp.domini.com	CNAME	domini.com
www.domini.com	CNAME	domini.com
pop.domini.com	CNAME	mail.proveidor.com
domini.com	MX	mail.proveidor.com

Alguns proveïdors de dominis també permeten modificar els registres de tipus SOA i NS, però en aquests, si es tenen altres serveis contractats amb el mateix proveïdor (per exemple, comptes de correu electrònic), poden deixar de funcionar.

2.3. Servei de directori (LDAP)

Els serveis de directori permeten enllaçar els recursos d'una xarxa amb les seues adreces. Mitjançant aquests serveis es poden gestionar, administrar i organitzar els elements i recursos de la xarxa, que inclouen carpetes, fitxers, usuaris i dispositius, entre altres objectes.

El protocol més utilitzat per treballar amb serveis de directoris és LDAP (*Lightweight Directory Access Protocol*). Aquest protocol permet iniciar una connexió segura, autenticar un usuari, realitzar cerques i comparacions, i gestionar entrades del directori.

Les entrades del directori són representades en el format *LDAP Data Interchange Format* (LDIF) com un arbre jeràrquic, i el contingut presenta l'aspecte següent:

```
dn: cn=John Doe,dc=example,dc=com
```

```
cn: John Doe
givenName: John
sn: Doe
telephoneNumber: +1 888 555 6789
telephoneNumber: +1 888 555 1232
mail: john@example.com
manager: cn=Barbara Doe,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

A partir d'aquesta entrada es poden identificar els següents elements a la primera línia:

- dn: és el nom distintiu de l'entrada (*distinguised name* (DN), en anglès). No és un atribut ni forma part de l'entrada.
- cn=John Doe: és el nom distintiu relatiu.
- dc=example,dc=com: és el nom distintiu de l'entrada pare.

La resta d'elements de l'entrada són atributs que proporcionen més informació sobre l'entrada, com el nom, l'adreça de correu electrònic, el número de telèfon, etc.

Habitualment, els serveis de directoris són utilitzats per organitzacions grans com les empreses públiques i les universitats. En aquests casos, els recursos no estan limitats a una xarxa local, sinó que pot ser que estiguen connectats a través d'internet.

Les operacions que pot realitzar un client sobre el servidor LDAP són les següents:

- **StartTLS**: per iniciar una connexió segura.
- **Bind**: per autenticar-se al servidor.
- **Search**: per cercar i accedir a entrades del directori.
- **Compare**: per comprovar si una entrada conté un valor d'atribut determinat.
- **Add**, **Delete** i **Modify**: per afegir, eliminar i modificar entrades, respectivament.
- **Modify Distinguished Name (DN)**: per moure o canviar el nom d'una entrada.
- **Abandon**: per cancel·lar una petició.
- **Extended operation**: per definir altres operacions.
- **Unbind**: per tancar la connexió.

Un dels avantatges de fer servir un servei de directori és que fa possible utilitzar el mateix nom d'usuari i contrasenya per autenticar un usuari en diferents recursos o, fins i tot, serveis aliens a la xarxa privada.

2.3.1. Configuració i administració d'usuaris en phpLDAPadmin

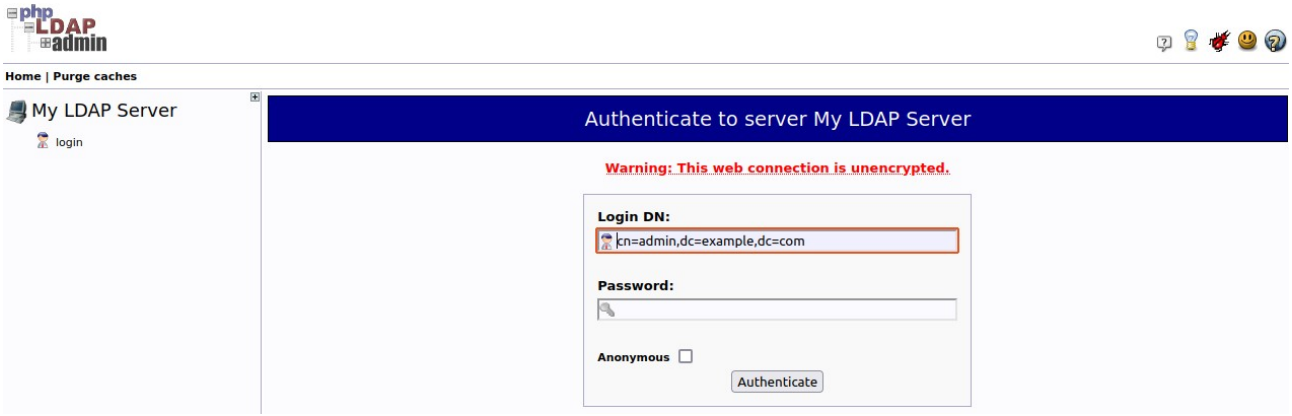
Com que gestionar el servei de directoris mitjançant la línia d'ordres és molt pesat, s'acostuma a gestionar-los utilitzant altres eines que proporcionen una interfície gràfica.

Entre aquestes eines hi ha phpLDAPadmin, un client LDAP web que permet administrar un servidor LDAP instal·lat a Linux mitjançant qualsevol navegador. Per instal·lar aquest client és necessari que el servidor tinga instal·lat prèviament:

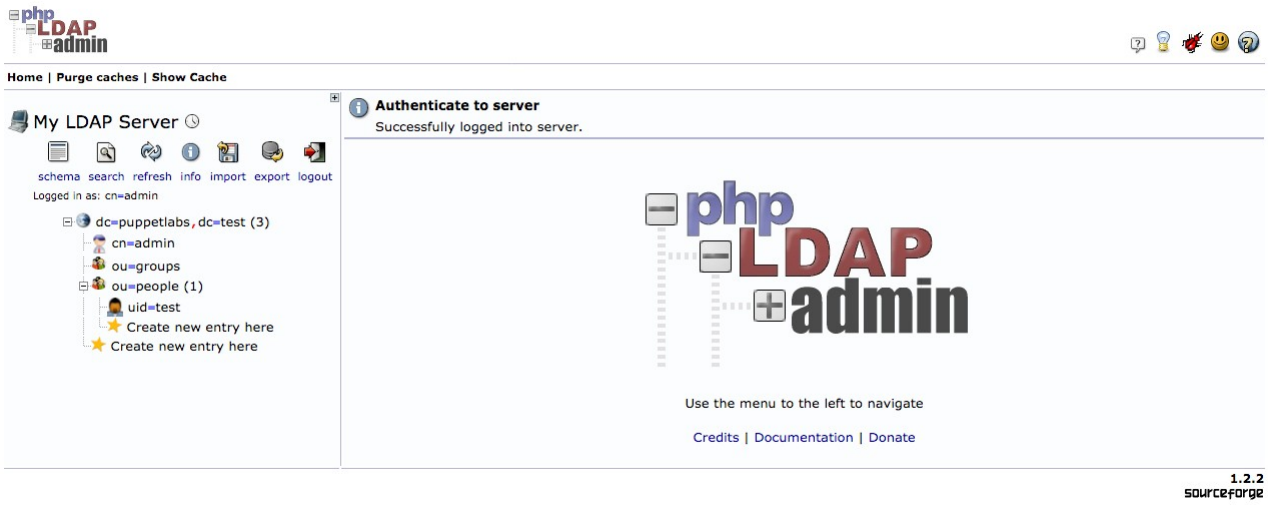
- Servidor LDAP
- Servidor web (per exemple, Apache)

- PHP v5 o superior que admeta les directives: PCRE, SESSION, GETTEXT, LDAP i XML.

Phpldapadmin té aquest aspecte:



Una vegada autenticats, a l'esquerra de la pantalla es mostra un arbre desplegable on es pot veure tota la informació del directori:



compte que el “Common Name” està format pel “First Name” i el “Last Name” i, per consegüent, s’omplirà automàticament.

El nom distingit d’un usuari es pot veure a la capçalera del detall de l’entrada corresponent.

Proveu de desconnectar-vos del compte “admin” i connecteu-vos amb el nou usuari que heu creat. Recordeu que si heu afegit l’entrada com a descendent d’una altra, s’ha d’afegir aquesta informació al nom distingit. Per exemple, en el cas d’haver creat l’entrada amb cn=ioc daw com a descendent de la branca ou=people, el nom distingit complet de l’usuari serà: cn=ioc daw,ou=people,dc=puppetlabs,dc=test.

2.3.2 Interacció dels serveis de directori amb aplicacions web

Donat que LDAP permet als usuaris connectar-se remotament i iniciar sessió, és possible desenvolupar aplicacions que facen servir aquesta funcionalitat per autenticar els usuaris.

Podeu trobar-ne un exemple a Jira (www.atlassian.com/software/jira), un programari de gestió de projectes utilitzat per grans organitzacions que pot configurar-se per autenticar els usuaris mitjançant múltiples serveis de directoris (LDAP, entre ells).

Un altre exemple d’autenticació es pot trobar a Microsoft Imagine, una plataforma de Microsoft que permet la descàrrega gratuïta i legal del seu programari a estudiants d’instituts i universitats (per a la resta d’usuaris els continguts són de pagament). En el cas d’alguns centres, com la Universitat Oberta de Catalunya, l’autenticació es realitza mitjançant un servei de directori que bloqueja els usuaris que no són estudiants o que no estan matriculats en el curs actual.

Pel que fa al desenvolupament d’aplicacions que utilitzen aquests serveis no és difícil trobar biblioteques que faciliten la tasca d’accedir a la informació proporcionada per LDAP en els llenguatges de programació més populars:

- Per a Node.js: LDAPJS (ldapjs.org)
- Per a Java: LDAP Java API (directory.apache.org/api/java-api.html)
- Per a PHP: no cal cap biblioteca, només cal activar-la a la configuració.
- Per a Ruby: Net::LDAP (goo.gl/wuZkmK)

Així doncs, en cas d’haver d’implementar l’autenticació mitjançant LDAP en qualsevol aplicació, només cal cercar una biblioteca (o les opcions de configuració) per al llenguatge utilitzat.

3. Utilització de serveis de xarxa i automatització

L’any 2009 es va crear Node.js, un entorn de programació per escriure aplicacions basat en el motor V8 de JavaScript de Chrome. Els seus usos més habituals són el desenvolupament de serveis web, els servidors d’aplicacions (com poden ser xats o servidors de jocs) i les eines per a la línia d’ordres. Per aquests motius, Node.js s’ha convertit en un component fonamental en el desenvolupament d’aplicacions modernes.

Com que JavaScript és un dels llenguatges fonamentals per a tot desenvolupador d'aplicacions web, aprendre a desenvolupar aplicacions amb Node.js i utilitzar les eines de la línia d'ordres és força senzill.

Aquestes eines faciliten molt la feina dels desenvolupadors i augmenten la productivitat, ja que permeten generar els fitxers optimitzats pel desplegament a partir dels fitxers de producció.

Entre moltes altres funcions, aquestes eines faciliten les tasques següents:

- Preprocessar fitxers Less i Sass per generar CSS.
- Compilar codi ES6 o TypeScript a ES5 (la versió de JavaScript que utilitzen la majoria dels navegadors).
- Copiar fitxers entre carpetes (per exemple, des de les carpetes de les biblioteques de desenvolupament a les carpetes públiques de l'aplicació per desplegar).

Tot i que aquestes eines no s'executen a la banda del client, la seva finalitat és processar el codi per generar la versió de l'aplicació web per desplegar.

Cal destacar que com que es tracta d'eines que funcionen completament mitjançant la línia d'ordres, poden fer-se servir en qualsevol entorn, com pot ser la terminal del sistema operatiu, un servidor remot al qual us connecteu utilitzant SSH o una màquina virtual gestionada per Vagrant.

Aquestes eines poden instal·lar-se de diferents maneres:

- Descarregant-les des d'una pàgina web.
- Important-les des d'un repositori de control de versions (com pot ser GitHub).3.13.
- Utilitzant un gestor de paquets com npm o Yarn.

Com podeu imaginar, cada cop que es fan canvis al codi, siga al codi JavaScript o CSS, s'han de repetir tots els processos lligats. Per exemple, compilar els fitxers en ES6 (si s'ha escollit aquesta versió del llenguatge), concatenar els fitxers JS generats per reduir el nombre de peticions i minimitzar el fitxer normal. En el cas de projectes grans, pot ser molt fàcil ometre algun pas o concatenar els fitxers en un ordre incorrecte.

La solució a aquest problema és utilitzar alguna eina d'automatització com GULP (https://gulpjs.com) o GRUNT(https://gruntjs.com), que permeten configurar i executar de forma ordenada totes les tasques necessàries. D'aquesta manera, només cal executar l'automatitzador per generar tots els fitxers necessaris per al desplegament.

3.1. Instal·lació de Node.js i npm

El primer pas per poder utilitzar qualsevol eina basada en Node.js és instal·lar Node.js. Podeu descarregar l'instal·lador des de la pàgina web corresponent: nodejs.org/es.

Cal destacar que en el cas d'algunes distribucions de Linux el programari Node.js pot venir preinstal·lat o pot instal·lar-se mitjançant un gestor de paquets (com APT), però aquesta versió acostuma a estar molt més endarrerida que les versions que es troben a la pàgina de Node.js.

Una vegada instal·lat a l'equip, per comprovar que la instal·lació s'ha portat a terme correctament, heu d'accedir al símbol del sistema o la terminal (segons el sistema operatiu) i introduir:

```
node -v
```

El nom del programa Node.js a Linux és nodejs, per evitar conflictes amb altres paquets.

npm

És, per defecte, el gestor de paquets per a Node.js (www.npmjs.com) .

La instal·lació de Node.js també acostuma a incloure el gestor de paquets npm, que permet actualitzar-lo i instal·lar biblioteques, eines i mòduls per utilitzar en els programes desenvolupats amb Node.js, com Express i Socket.io. Tot i així, si feu servir un gestor de paquets com APT per fer la instal·lació de Node.js, haureu de fer la instal·lació de npm de forma independent.

La instal·lació a Linux mitjançant els gestors de paquets pot ser problemàtica, ja que habitualment no tenen les versions més recents de Node.js ni npm als seus repositoris.

3.1.1. Creació d'un servidor HTTP amb Node.js

Per comprovar el funcionament de Node.js, podeu provar el programa següent. Consisteix en la creació d'un servidor HTTP que respon a qualsevol petició amb el missatge "Hola món" en format text pla.

Creeu un fitxer anomenat http-server.js, amb el contingut següent:

```
var http = require('http');
http.createServer(function(peticio, resposta) {
        resposta.writeHead(200, {'Content-Type':'text/plain;charset=utf-
8'});
        resposta.end('Hola món');
}).listen(8080, '127.0.0.1');
console.log('Servidor executant-se a http://127.0.0.1:8080/');
```

Executeu-lo, escrivint a la línia d'ordres: `node http-server` (en linux `nodejs http-server`).

Els programes desenvolupats amb Node.js s'anomenen mòduls.

Aquest codi realitza les accions següents:

- Importa el mòdul http.
- Crea un servidor que escolta pel port 8080 i l'adreça IP 127.0.0.1.
- La resposta del servidor a totes les peticions és la següent: escriu a la capçalera el codi 200, defineix el tipus de contingut com a text pla amb el joc de caràcters UTF-8 i afegeix com a cos de la resposta el missatge "Hola món".
- Mostra un missatge a la terminal per indicar a l'usuari que s'ha iniciat el servidor.

Si obriu l'URL 127.0.0.1:8080 al vostre navegador, veureu que es mostra una pàgina en blanc amb el missatge "Hola Món".

Com es pot apreciar, és molt senzill crear un servidor amb Node.js, perquè es tracta d'un entorn de programació orientat a la creació de serveis web.

3.2. Gestors de paquets

A l'hora d'instal·lar una biblioteca o mòdul per utilitzar a les aplicacions, disposeu de diverses opcions, però una de les més recomanables és utilitzar un gestor de paquets especialitzat en el desenvolupament d'aplicacions web.

L'avantatge d'utilitzar els gestors de paquets és que la informació d'aquestes biblioteques passa a ser gestionada pel gestor de paquets. Això facilita la distribució de l'aplicació i l'actualització de les dependències, ja que a partir del fitxer de configuració (per exemple, el fitxer package.json per a npm) podeu tornar a descarregar tots els fitxers necessaris per al projecte. A més, els gestors s'encarreguen de descarregar també totes les dependències. És a dir, si una biblioteca requereix cinc mòduls diferents, aquests mòduls es descarreguen automàticament.

Entre els gestors de paquets més utilitzats hi ha **npm** i **Yarn**. npm va ser creat poc després de l'aparició de Node.js, té més de 300.000 paquets enregistrats i és utilitzat per més de 5 milions de desenvolupadors. Per la seva banda, Yarn va ser desenvolupat pels enginyers de Facebook en col·laboració amb els enginyers d'Exponent, Google i Tilde.

npm

Podeu trobar més informació sobre el gestor de paquets npm a l'enllaç següent: www.npmjs.com.

npm és el gestor de paquets més utilitzat i es troba inclòs a la instal·lació de Node.js. Per comprovar que es troba instal·lat correctament al vostre equip podeu escriure a la línia d'ordres: `npm -v`

Aquest gestor permet instal·lar nous mòduls, biblioteques i entorns de treball a Node.js. Per exemple, per instal·lar la biblioteca Express per crear un servidor HTTP avançat, es fa amb l'ordre següent: `npm install express`

En executar aquesta ordre, es descarrega la biblioteca Express juntament amb totes les seves dependències, que queden emmagatzemades dins d'una carpeta a l'arrel del projecte anomenada *node_modules*.

A més a més, pot actualitzar-se a si mateix amb l'ordre: `npm install npm@latest -g`

Fixeu-vos en el paràmetre -g. Indica que aquesta ordre s'ha d'executar per a la instal·lació global de npm. Utilitzant aquest mateix paràmetre poden instal·lar-se paquets globalment. Això és especialment útil a l'hora d'instal·lar eines per a la línia d'ordres.

Vegeu com s'instal·la **Babel**, el compilador de TypeScript i ES6: `npm install babel-cli -g`

Yarn

Yarn (yarnpkg.com) pot instal·lar-se mitjançant npm però no es recomana: la instal·lació no es realitza de forma determinista, el paquet no és signat i només es fa una comprovació d'integritat, com a mesura de seguretat. En conseqüència, els desenvolupadors de Yarn estimen que aquest tipus d'instal·lació és un risc de seguretat per a aplicacions grans.

Tot i així, en cas de voler instal·lar-lo utilitzant npm, l'ordre seria la següent:

```
npm install yarn -g
```

Yarn va ser desenvolupat per Facebook per solucionar els problemes que tenien en utilitzar npm com a gestor de paquets. Quan treballaven en projectes grans, es trobaven que per a cada canvi en algun dels fitxers es produïen enviaments de canvis al repositori amb centenars de milers de línies (que s'havien regenerat automàticament), i per solucionar els problemes produïts es perdia un dia de feina d'un dels enginyers.

Per altra banda, no hi havia cap garantia que les dependències es descarraren en el mateix ordre en tots els equips, de manera que podien produir-se inconsistències: segons l'ordre de descàrrega, la versió d'alguna dependència podia canviar (per exemple, perquè es tractava d'una subdependència).

A més a més, el nombre de fitxers descarregats per npm en alguns projectes era exageradament gran. Per exemple, en instal·lar React Native, que tenia 62 dependències, s'havien de descarregar, a la carpeta *node_modules*, més de 120.000 fitxers.

Per resoldre aquests problemes, Facebook, juntament amb Exponent, Google i Tilde, va crear Yarn com un gestor de paquets més ràpid i fiable del qual destaquen les característiques següents:

- Cau fora de línia: Yarn descarrega, al disc de l'equip, una còpia de cada paquet utilitzat i es diferencien per registre d'origen (per exemple, npm) i versió (per exemple, 4.1.4). D'aquesta manera, es poden fer instal·lacions fora de línia i només es descarreguen els paquets que no es trobin al cau.
- Instal·lacions deterministes: la instal·lació dels paquets en qualsevol equip sempre es du a terme en el mateix ordre i amb la mateixa versió dels paquets.
- Ràpid: la instal·lació de paquets és més rapida perquè es realitza en paral·lel i fa servir el cau per als paquets que s'han descarregat prèviament.
- Segur: abans d'executar el codi de qualsevol paquet Yarn, realitza una comprovació d'integritat del fitxer.

Tot i que pot semblar que aquest gestor és superior a npm, sembla que en alguns casos no funciona correctament. Respecte a les instal·lacions deterministes, es poden obtenir a npm mitjançant l'opció shrinkwrap o indicant al fitxer package.json la versió exacta que es vol utilitzar.

3.3. Preprocessadors de CSS

Els fulls d'estil en cascada (en anglès, *cascading style sheets* o CSS) són un element indispensable per canviar la representació de qualsevol pàgina o aplicació web, ja que aquest llenguatge és el responsable de modificar l'aspecte que mostren els elements HTML. Gràcies al CSS, es pot modificar la font utilitzada en un text, la mida o el color, indicar el tipus de vora que es vol afegir, etc. En les versions més recents, fins i tot es pot animar qualsevol element HTML.

Malauradament, CSS és un llenguatge de marques i no inclou funcionalitats avançades com la utilització de variables, operadors i funcions definides pels usuaris. Això provoca greus problemes de manteniment, ja que en qualsevol lloc web és habitual haver de

reutilitzar elements i configuracions (per exemple, el color principal de l'empresa, l'estil de les ombres o el radi d'una vora arrodonida).

Exemple de canvi del color principal al lloc web d'una empresa

En el lloc web d'una empresa s'ha estat fent servir el color roig (#ff00000) com a color principal de l'empresa (corresponent al de la seva marca) per als botons, els enllaços, les capçaleres i el text destacat.

Un dia descobreixen que això no és correcte, que el color principal de la seua marca és (#ff2020) i s'ha de canviar a tot. Això implica modificar tots els fitxers CSS i fer un reemplaçament on s'haga fet servir el color, però només quan es mostra com a color de la marca (per exemple, sense modificar el color per a mostrar errors).

Com que aquest procés és manual, és molt possible cometre errades: deixar fitxers sense modificar, modificar colors que no mostren elements de la marca o modificar missatges d'error amb el nou color.

Un cas més extrem podria consistir a canviar els colors d'un entorn de treball complet, com per exemple Bootstrap, per ajustar-lo al de l'empresa. Això implicaria canviar no sols el color principal, sinó també els múltiples colors que es poden trobar a l'entorn, que són lleugerament més clars o més foscos que el color principal. Aquest tipus de canvi requeriria modificar centenars de línies de codi (a Bootstrap hi ha més de 500 elements amb propietats que modifiquen el color).

Per solucionar aquests problemes es van crear els preprocessadors de CSS.

Els preprocessadors de CSS fan servir un llenguatge propi que augmenta el llenguatge CSS. Després, es preprocessen aquests fitxers i es generen els fitxers amb el codi CSS. Entre les característiques que es poden trobar en aquests llenguatges hi ha les següents:

- **Declaració de variables**: permeten emmagatzemar valors com colors o dimensions (amplada, alçària, mida de font, etc.).
- **Operadors**: permeten realitzar operacions entre nombres o variables.
- ***Mixins***: permeten afegir blocs de codi CSS configurats utilitzant arguments (de forma similar a les funcions).
- **Funcions predefinides**: cada preprocessador inclou les seves, però és habitual trobar la funció lighten, que permet aclarir un color, o darken, que permet enfosquir-lo.
- **Importació**: permet importar altres fitxers, de manera que és més fàcil organitzar-los. Per exemple, el fitxer "principal" pot importar els fitxers "variables", "colors", "mides" i "pagina". El resultat de preprocessar "principal" serà un únic fitxer CSS amb el contingut dels 5 fitxers.
- **Imbricació** (en anglès, *nesting*): permet imbricar el codi generant una jerarquia en forma d'arbre, de manera que el codi generat és més senzill d'entendre i de modificar.
- **Condicionals i bucles**: permeten la creació de bucles i aplicar unes regles o unes altres de forma condicional.

Cal destacar que no totes les noves característiques que formen part de l'especificació de CSS es troben implementades a tots els navegadors. En alguns casos es requereix utilitzar un prefix especial perquè encara es troben en fase experimental o pot ser que no siguin implementades. Per aquesta raó, alguns preprocessadors inclouen automàticament les versions prefixades de les regles més recents i, quan és possible, alguna alternativa per als navegadors més antics.

S'ha de tenir en compte que segons quin preprocessador s'utilitza es pot treballar directament amb codi CSS estàndard o no, ja que no tots l'admeten.

Els preprocessadors més populars són els següents:

- **Sass**: és més potent que Less però més complicat d'aprendre, tot i que amb la nova sintaxi és molt més similar a Less.
- **Less**: inclou menys característiques que Sass, tot i que les més importants es troben presents i és més fàcil d'aprendre perquè la seva sintaxi és molt similar a la de CSS.
- **Stylus**: és un preprocessador inspirat en Less i Sass que inclou la majoria de les característiques dels dos i també altres de pròpies. És el més complex dels tres, però també ofereix més opcions en tots els àmbits.

Com que Sass i Less fa més temps que circulen, és habitual trobar que els entorns de treball i biblioteques ofereixen, a banda dels fitxers CSS, els fitxers en format Sass o Less, perquè es puguen fer les pròpies compilacions. Per exemple, Bootstrap ofereix l'opció de descarregar els fitxers en tots dos formats, de manera que canviar el color principal de l'entorn només requereix canviar una única variable i preprocessar-lo.

3.3.1. Sass

Sass (sass-lang.com) és un projecte de codi lliure que ha estat en desenvolupament durant més de deu anys i ha assentat les bases dels preprocessadors CSS moderns.

Actualment accepta dues sintaxis diferents:

- **Sass** (extensió ".sass"): té la sintaxi original, no és directament compatible amb CSS.
- **SCSS** (extensió ".scss"): té una nova sintaxi més similar a CSS i Less.

Convé tenir en compte que la sintaxi Sass no és compatible directament amb CSS, al contrari que SCSS. És a dir, si el contingut d'un fitxer Sass és codi CSS, es produeix un error en preprocessar-lo; en canvi, amb SCSS és preprocessa sense problema.

Sass inclou les següents característiques: variables, imbricació, importació i parcials, *mixins*, herència, operadors i funcions predefinides.

Les **variables** s'identifiquen perquè van prefixades pel símbol $. Per exemple, per establir el color i la mida de la font utilitzant variables es fa de la manera següent:

```
$mida-font: 15px;
$color: #ff0000;

p {
```

```
  font-size: $mida-font;
  color: $color;
}
```

I, en ser preprocessat, generaria la sortida següent:

```
p {
  font-size: 15px;
  color: #ff0000;
}
```

La **imbricació** permet imbricar elements dins d'altres elements.

En el cas d'una barra de navegació (element nav), es poden afegir els estils de tots els elements que s'apliquen només quan aquests es troben dintre de la barra de navegació:

```
nav {
  ul {
    list-style: none;
    margin: 0;
    padding: 0;
    color: light-grey;
  }

  li {
    display: inline-block;
  }

  a {
    text-decoration:none;
    display: block;
    padding: 5px;
  }
}
```

Com es pot apreciar, es veu molt clarament com s'organitzen els estils. Vegeu, a continuació, el codi generat en preprocessar-lo:

```
nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
  color:light-grey;
}

nav li {
  display: inline-block;
}

nav a {
  text-decoration: none;
  display: block;
  padding: 5px;
}
```

En aquest cas no hi ha gaire diferència, però es perd la claredat que proporciona visualitzar els continguts jerarquitzats.

Sass permet **importar fitxers sencers o parcials** mitjançant la directriu @import. La diferència es troba en el fet que els fitxers parcials no són preprocessats en fitxers

individuals, ja que són inclosos en altres fitxers. El nom dels fitxers parcials sempre ha de començar per una barra baixa (_).

3.3.2. Less

Less (lesscss.org) és un preprocessador que pot executar-se a Node.js, al navegador (encara que només es recomana durant el desenvolupament) i a Rhino. Cal tenir en compte que, en conjunt, Sass és més complet que Less, però tant l'un com l'altre tenen les característiques bàsiques més importants.

A Less, s'hi poden trobar les característiques següents: variables, imbricació, importació, *mixins*, herència, operadors i funcions predefinides. A Less, per declarar **variables** s'utilitza el símbol @.

Exemple de definició de variables a Less

Vegeu com es defineixen dues variables per establir el color i la mida de la font dels elements de tipus paràgraf:

```
@mida-font: 15px;
@color: #ff0000;

p {
  font-size: @mida-font;
  color: @color;
}
```

El codi CSS preprocessat és el següent:

```
p {
  font-size: 15px;
  color: #ff0000;
}
```

La **imbricació** d'elements és idèntica a com es codifica a Sass (amb la sintaxi SCSS). Per imbricar els estils necessaris per formatar una barra de navegació, el codi Less és el següent:

```
nav {
  ul {
    list-style: none;
    margin: 0;
    padding: 0;
    color: light-grey;
  }

  li {
    display: inline-block;
  }

  a {
    text-decoration:none;
    display: block;
    padding: 5px;
  }
}
```

I el resultat CSS és el que trobeu a continuació:

```
nav ul {
```

```
  list-style: none;
  margin: 0;
  padding: 0;
  color: light-grey;
}
nav li {
  display: inline-block;
}
nav a {
  text-decoration: none;
  display: block;
  padding: 5px;
}
```

Less també inclou la capacitat d'importar altres fitxers que, en ser processats, són inclosos directament al fitxer CSS. Per fer-ho, s'utilitza la directriu @import, igual que a Sass, com es pot comprovar en l'exemple següent:

```
@import 'parcial';
@import 'colors.css';
```

Si no s'afegeix cap extensió al fitxer Less, s'interpreta que l'extensió ha de ser .less. En qualsevol cas (a excepció de l'extensió .css, que no és preprocessada), s'interpreta el fitxer com a codi Less.

3.3.3. Stylus

Stylus (stylus-lang.com) és un preprocessador que inclou totes les característiques de Sass i Less (està inspirat en tots dos) però també hi afegeix les seves pròpies característiques. Com que la seva sintaxi és força diferent de la de Sass i Less i ofereix gran quantitat d'opcions, Stylus és un preprocessador més difícil de dominar.

Una diferència important amb altres preprocessadors és que els punts i comes, les comes i els claudàtors són opcionals. Per altra banda, el sagnat ha de fer-se correctament, ja que en cas contrari el resultat no seria l'esperat. Una altra diferència és que les variables a Stylus poden declarar-se al mateix lloc que es fan servir i no requereixen cap prefix especial (tot i que es pot fer servir el símbol $).

Pel que fa a les funcions, Stylus no només té funcions predefinides sinó que permet crear funcions pròpies que retornin valors i no només mostrin codi CSS. El conjunt d'operadors admesos és molt superior que el d'altres processadors, ja que inclou operadors binaris, unaris, lògics i fins i tot admet rangs de llistes.

Stylus s'instal·la mitjançant el gestor de paquets npm amb l'ordre següent:

```
npm install stylus -g
```

Una vegada instal·lat, es pot utilitzar l'ordre stylus indicant el nom del fitxer amb el codi Stylus i el nom del fitxer CSS de destí. Per comprovar-ho, creeu un fitxer anomenat prova.styl amb el contingut següent:

```
color_principal= blue
color_vora = darken(color_principal, 20%)
color_enllac = lighten(color_principal, 20%)
```

```
div
  color: color_principal
  border: 1px solid color_vora

  a
    color:color_enllac
```

Fixeu-vos que, per assignar el valor a les variables, s'ha fet servir el signe igual (=) i el nom de les variables s'ha separat amb una barra per sota en lloc d'un guió (a Stylus no es pot fer servir el guió en els noms de variables).

Per generar el fitxer CSS escriviu a la línia d'ordres:

```
stylus prova.styl resultat.css
```

Es generarà el fitxer resultat.css a la mateixa carpeta amb el contingut següent:

```
div {
  color: #00f;
  border: 1px solid @00c;
}
div a {
  color: #33f;
}
```

3.4. Compiladors JavaScript

Tot i que la idea d'un compilador per a un llenguatge interpretat com a JavaScript pot semblar estranya, es tracta d'una eina molt utilitzada.

Un dels problemes de JavaScript és que quan s'executa en el navegador d'usuari, no es pot controlar l'entorn d'execució. Pot tractar-se d'un navegador en un ordinador d'escriptori o d'un dispositiu mòbil, i no hi ha cap garantia que aquest navegador estigui actualitzat, ni tan sols que el seu fabricant hagi inclòs les característiques necessàries per al bon funcionament de la vostra aplicació.

S'ha de tenir en compte que el ritme d'adaptació dels navegadors a les noves versions de JavaScript és molt lent. Per exemple, ES5 va trigar 10 anys a ser completament admès per la majoria dels navegadors, és a dir, el mateix any que va ser llançada la següent versió.

En cas de voler utilitzar les versions més recents de JavaScript és possible implementar les aplicacions utilitzant aquesta versió, i posteriorment compilar-la a una versió que siga admesa més àmpliament. En aquest àmbit, el compilador més utilitzat es Babel.js, que permet compilar programes desenvolupats en ES2015 i posteriors a ES5.

Un altre compilador de JavaScript (no molt utilitzat) és el Closure Compiler (goo.gl/4AR0x5), desenvolupat per Google, que permet optimitzar el codi i, mitjançant els comentaris inclosos com a documentació en format JSDOC, inspeccionar determinats comportaments que en cas que no es complissin, mostrarien avisos durant la compilació. Per exemple, si s'anota una variable com si fos una constant i s'intenta canviar el valor més endavant, es mostra un avís durant la compilació.

Per altra banda, al llarg del temps s'han creat múltiples llenguatges que són compilats a JavaScript. D'aquesta manera, poden ser desenvolupats en aquests llenguatges i ser

utilitzats al navegador (recordeu que l'únic llenguatge que pot executar-se en aquest entorn és JavaScript). Entre aquests llenguatges hi ha CofeeScript, TypeScript i Dart.

Fora de l'entorn dels navegadors cal destacar el compilador Rhino, utilitzat per compilar JavaScript a Java, que s'utilitza incrustat en altres dispositius i permet programar-los utilitzant JavaScript.

3.4.1. ECMAScript 2023 8ES14) i Babel

Quan es parla d'ES14 es fa referència a la sisena edició de JavaScript, tot i que el nom oficial de l'estàndard és **ECMAScript 2023**. És a dir, JavaScript és la implementació de l'estàndard ECMAScript.

Cada any, el mes de juny, es publica una ampliació de l'especificació i es canvia el nom.

Quan es treballa amb alguna característica avançada del llenguatge, és important saber en quina versió s'ha afegit per poder determinar si està disponible o no, ja que algunes característiques només existeixen com a especificació i no es troben implementades en cap navegador ni compilador.

Tot i que moltes de les característiques no es troben implementades als navegadors, hi ha alguns desenvolupadors que prefereixen utilitzar aquestes versions del llenguatge, ja que eventualment aquest és el llenguatge que cal utilitzar al web. A més, aquestes versions ofereixen moltes característiques interessants, com ara la creació de classes, la declaració de constants, el context de bloc, la importació de mòduls, l'encapsulació i la utilització de funcions fletxa o lambdes.

El compilador més utilitzat per compilar el codi ECMAScript és Babel (babeljs.io). Per instal·lar-lo mitjançant el gestor de paquets npm podeu utilitzar l'ordre següent:

```
npm install --save-dev babel-cli babel-preset-env
npm install --save-dev babel-preset-latest
```

Un cop instal·lats aquests paquets, podeu començar a utilitzar Babel per compilar JavaScript.

3.4.2. TypeScript

TypeScript (www.typescriptlang.org) és un llenguatge de programació desenvolupat i mantingut per Microsoft basat en l'especificació ECMAScript al qual afegeix (opcionalment) tipificació estàtica, és a dir, es comprova que els tipus de variables i paràmetres siguen correctes durant la compilació.

Tot i que la tipificació estàtica és opcional, és molt recomanable utilitzar-la, ja que ajuda a prevenir errors que d'altra manera poden passar desapercebuts fàcilment.

3.5. Eines d'automatització

Quan es desenvolupen aplicacions web és molt important optimitzar els fitxers generats, perquè es redueix el pes i el nombre de peticions HTTP que s'han de realitzar per carregar l'aplicació.

Els navegadors només processen entre 4 i 8 peticions HTTP al mateix temps, i el temps mitjà per obtenir la resposta a cada petició és al voltant dels 100 ms en línies d'alta velocitat (ADSL, fibra i cable) i per sobre de 400 ms en línies mòbils (3G, 4G), independentment del pes de la resposta. És a dir, en molts casos el coll d'ampolla per descarregar els continguts d'una pàgina web es troba en el nombre de peticions i no pas en el pes dels continguts.

Aquest és un dels problemes més greus d'utilitzar sistemes gestors de continguts (CMS) com WordPress o Joomla, ja que acostumen a realitzar 60 peticions (superant-ne el centenar en alguns llocs) només per descarregar fitxers JS i CSS. Aquest nombre de peticions es podria reduir a 2 o 3 fitxers en la majoria dels casos.

Convé destacar que, molt sovint, quan es parla de desenvolupar un lloc o una aplicació web, s'acostuma a pensar en un nombre molt xicotet de fitxers: un parell de fitxers CSS, un fitxer amb el codi JavaScript, un fitxer HTML i algunes imatges decoratives. Però avui dia això està molt lluny de la realitat, atesa la complexitat de les aplicacions que s'han de desenvolupar, les dependències necessàries (compiladors de JavaScript, preprocessadors de CSS, gestió de fitxer, etc.) i el nombre i tipus de fitxers diferents amb els quals s'ha de treballar.

A més, s'ha de tenir en compte que molts d'aquests fitxers no formen part del desplegament i es generen molts fitxers intermedis. Per exemple, en el cas del preprocessament de fitxer Less es podrien fer els passos següents:

1. Es preprocessen els fitxers Less i es generen els fitxers CSS.
2. Els fitxers CSS es concatenen per formar un únic fitxer.
3. Es minimitza el fitxer final.

Cal ressaltar que, si canvia l'ordre de concatenació dels fitxers, també pot canviar el resultat final (els estils CSS s'apliquen en cascada), així que és necessari que el procés es realitzi sempre en el mateix ordre. Per altra banda, al desplegament només cal afegir-hi el fitxer final minimitzat; la resta de fitxers CSS es poden descartar, i els fitxers Less no han de sortir de l'entorn de desplegament.

Fixeu-vos que aquest cas s'aplica als fitxers Less propis, però segurament haureu d'incloure en aquest procés els fitxers Less i CSS inclosos en altres biblioteques. Per exemple, si el projecte inclou Bootstrap i FontAwesome, aquests també s'han d'afegir al vostre fitxer CSS:

Variables Less de Bootstrap

1. En el cas de Bootstrap, s'ha de descarregar la versió Less i importar-la al vostre fitxer Less principal per poder modificar directament les variables i processar-lo juntament amb els vostres fitxers.
2. El fitxer CSS de FontAwesome es pot afegir directament al començament de la concatenació de fitxers CSS.

Continuant amb el mateix exemple, el següent pas és copiar els fitxers d'imatges i fonts necessàries a la carpeta de desplegament, tenint en compte que aquestes imatges i fonts

poden canviar al llarg del desenvolupament, ja sigui perquè s'han modificat algunes imatges o perquè s'ha actualitzat la biblioteca que les incloïa.

Després, s'hi han d'afegir els fitxers JavaScript, seguint un procés similar al dels fitxers Less:

1. Compilar els fitxers necessaris (ES2015, TypeScript, Dart, etc.) a ES5.
2. Concatenar tots els fitxers JavaScript en un sol fitxer en el següent ordre:
    1. Biblioteques principals utilitzades per l'aplicació (per exemple, jQuery, AngularJS, Vue.js, etc.)
    2. Codi d'altres biblioteques i connectors necessaris
    3. Codi propi
3. Minimitzar el fitxer resultant.

En alguns casos, cal generar els fitxers optimitzats amb diferents continguts (és a dir, repetir tots aquests processos amb diferents configuracions).

Per exemple:

- Per obtenir un fitxer que només inclogui el codi de la secció d'administració del lloc i que no serà carregat per usuaris no autoritzats.
- Per generar un fitxer amb els continguts comuns a totes les pàgines del lloc (que després pot ser concatenat a altres fitxers més especialitzats).
- Per generar fitxers únics per a diferents tipus de continguts de l'aplicació: uns fitxers per a la pàgina principal, un altre per a la pàgina de cerca, un altre per al panell d'administració, etc.

Com es pot apreciar, aquest procés és molt feixuc, pot consumir molt de temps i és molt fàcil de cometre errors. A més, s'ha de repetir cada vegada que es fan canvis al codi, cosa que fa inviable realitzar totes aquestes operacions manualment. Durant molts anys, s'han fet servir eines per automatitzar els processos en altres llenguatges: Ant, Graddle o Maven.

A JavaScript, s'ha aprofitat la potència de Node.js per crear eines pròpies que s'integren perfectament amb el flux de treball, ja que es programen i configuren amb JavaScript i permeten realitzar tot tipus de tasques. Les dues eines més populars per portar a terme aquesta automatització són: Grunt i Gulp.

La decisió de fer servir qualsevol d'aquestes eines generalment recau en la decisió del vostre equip de desenvolupament o de l'entorn de programació escollit. Per exemple, les aplicacions desenvolupades amb Laravel feien servir Gulp com a automatitzador, però a les versions més recents ha estat reemplaçat per Webpack (un gestor de paquets).

3.5.1. Gulp

Gulp (gulpjs.com) és una eina basada en Node.js que permet automatitzar el processament de tasques. La configuració es realitza mitjançant un fitxer que conté el codi JavaScript, en el qual s'afegeixen els mòduls necessaris i la configuració de les diferents tasques.

Per instal·lar Gulp mitjançant el gestor de paquets npm, s'han d'introduir les ordres següents:

```
npm install gulp-cli -g
npm install gulp -D
```

La primera ordre instal·la el client de Gulp de forma global, mentre que la segona instal·la el mòdul gulp i l'afegeix al fitxer package.json com a dependència de desenvolupament.

Gulp requereix un fitxer anomenat gulpfile.js. Aquest fitxer ha de contenir un programa de Node.js que serà utilitzat pel client de Gulp per portar a terme les tasques, a part de ser el responsable d'importar els connectors necessaris. Cal tenir en compte que els connectors s'han d'instal·lar per separat. Així, si voleu preprocessar fitxers Less i minimitzar-los, cal instal·lar primerament els connectors "gulp-less", "gulp-csso" i "gulp-concat", tal com podeu veure a continuació:

```
npm install gulp-less gulp-csso gulp-concat -g
```

Un cop instal·lats els connectors necessaris, es pot crear una tasca que processi tots els fitxers Less d'una carpeta, que concateni el fitxer CSS resultant i el minimitzi. El contingut de gulpfile.js seria similar al següent:

```
var gulp = require('gulp');
var less = require('gulp-less');
var minifyCSS = require('gulp-csso');
var concat = require('gulp-concat');

gulp.task('css', function(){
  return gulp.src('less/*.less')
    .pipe(less())
    .pipe(concat('estils.css'))
    .pipe(minifyCSS())
    .pipe(gulp.dest('build/css'))
});

gulp.task('default', [ 'css' ]);
```

Com podeu apreciar, Gulp fa servir un sistema de canonades: la sortida d'una funció és l'entrada de la següent. Així doncs, la sortida de la funció src és l'entrada de la funció less, la seva sortida, al seu torn, és l'entrada de la funció concat, i així successivament.

Per comprovar que funciona correctament, creeu un directori anomenat *less*, amb dos fitxers anomenats full1.less i full2.less.

Afegiu al fitxer full1.less el codi següent:

```
@mida-font: 15px;
@color: #ff0000;

p {
  font-size: @mida-font;
  color: @color;
}
I al fitxer full2.less, afegiu-hi el següent:
@color-principal: blue;
@color-vora: darken(@color-principal, 20%);
```

```
@color-enllac: lighten(@color-principal, 20%);

div {
  color: @color-principal;
  border: 1px solid @color-vora;

  a {
    color:@color-enllac;
  }
}
```

A continuació, executeu l'ordre següent:

gulp

L'eixida serà similar a la següent:

```
[10:33:06] Using gulpfile ~/gulpfile.js
[10:33:06] Starting 'css'...
[10:33:06] Finished 'css' after 84 ms
[10:33:06] Starting 'default'...
[10:33:06] Finished 'default' after 23 µs
```

Quan s'executa Gulp sense cap paràmetre, s'executa la tasca per defecte (default), que en aquest cas només inclou la tasca css i, per consegüent, el resultat és l'execució d'aquesta tasca.

El resultat és la creació d'un fitxer anomenat estils.css dintre del directori *build/css*. Si algun d'aquests directoris no existeix, es crea automàticament. El contingut d'aquest fitxer és el següent: preprocessat, concatenat i minimitzat.

La sortida serà similar a la següent:

```
p{font-size:15px;color:red}div{color:#00f;border:1px solid #009}div a{color:#66f}
```

Si s'executa Gulp indicant el nom de la tasca css, el resultat és el mateix:

```
gulp css
```

Fixeu-vos que en aquest cas l'eixida només indica que s'ha iniciat la tasca css:

```
[10:41:28] Using gulpfile ~/gulpfile.js
[10:41:28] Starting 'css'...
[10:41:28] Finished 'css' after 89 ms
```

Per afegir noves tasques al fitxer gulpfile.js, només cal invocar la funció task del mòdul gulp passant com a primer argument el nom de la tasca i com a segon argument una funció que conté el codi necessari per executar-la. Per exemple, per afegir una tasca que mostri un missatge per la pantalla només cal afegir al fitxer gulpfile.js el següent codi:

```
gulp.task('hola', function() {
  console.log("Hola món!");
});
```

Seguidament podreu executar la tasca "hola" amb l'ordre següent:

```
gulp hola
```

I el resultat mostrat per la pantalla serà el següent:

```
[10:49:08] Using gulpfile ~/gulpfile.js
[10:49:08] Starting 'hola'...
Hola món!
[10:49:08] Finished 'hola' after 477 µs
```

Fixeu-vos que si no hi afegiu cap paràmetre, l'execució per defecte no processa aquesta nova tasca, ja que no s'hi ha inclòs. Per afegir-la, reemplaceu la tasca default per la següent:

```
gulp.task('default', [ 'css', 'hola' ]);
```

Si ara executeu l'ordre gulp sense cap paràmetre, el resultat serà similar al següent:

```
[10:52:23] Using gulpfile ~/gulpfile.js
[10:52:23] Starting 'css'...
[10:52:23] Starting 'hola'...
Hola món!
[10:52:23] Finished 'hola' after 620 µs
[10:52:24] Finished 'css' after 92 ms
[10:52:24] Starting 'default'...
[10:52:24] Finished 'default' after 31 µs
```

Convé destacar un detall molt important: les tasques s'inicien paral·lelament, és a dir, totes les tasques s'inicien alhora i la segona tasca finalitza abans que la primera, perquè, com que és molt simple, requereix menys temps de processament. És un avantatge respecte a Grunt, ja que aquest sistema és més ràpid que executar les tasques seqüencialment.

Per altra banda, s'ha de tenir molt clar que cada tasca ha de ser independent de la resta per evitar conflictes. Per exemple, si teniu una tasca que concatena els fitxers CSS i una altra que els preprocessa, la tasca de preprocessament començarà abans que els fitxers siguin concatenats i el resultat serà incorrecte.

Val la pena recordar que gulpfile.js és un fitxer de JavaScript executat sobre Node.js i, conseqüentment, podeu utilitzar qualsevol dels seus mòduls segons les necessitats. Per exemple, seria possible utilitzar el mòdul FileSystem per desar la data i hora de l'última compilació, iniciar una bateria de proves unitàries o generar la documentació actualitzada.

3.5.2. Grunt

Grunt (gruntjs.com) és una de les primeres eines per automatitzar el flux de treball que es va desenvolupar sobre Node.js. Actualment té més de 5.000 connectors, fet que fa que ja existeixi el connector adequat per a, pràcticament, qualsevol tasca.

La principal diferència amb Gulp és que en lloc d'implementar les tasques mitjançant codi, aquest està basat en configuracions (tot i que el fitxer de configuració també és un fitxer en JavaScript).

Per instal·lar Grunt mitjançant el gestor de paquets npm escriviu les ordres següents:

```
npm install grunt-cli -g
npm install grunt -D
```

Com es pot apreciar, el client de Grunt s'instal·la globalment, mentre que el mòdul Grunt s'instal·la localment i s'afegeix com a dependència de desenvolupament al fitxer package.json.

A continuació, s'han d'instal·lar els connectors necessaris. Per exemple, si es vol crear una tasca per preprocessar fitxers Less, concatenar-los i minimitzar-los, cal instal·lar els connectors "grunt-contrib-cssmin" i "grunt-contrib-less" (aquest connector s'encarrega, a més a més, de fer la concatenació automàticament):

```
npm install grunt-contrib-cssmin --save-dev
npm install grunt-contrib-less --save-dev
```

Una vegada instal·lats els connectors, podeu crear el fitxer Gruntfile.js amb el contingut següent:

```
module.exports = function(grunt) {

  grunt.initConfig({
    less: {
        dist: {
          src: ['less/*.less'],
          dest: 'build/css/estils.css'
        }
    },

    cssmin: {
      target: {
        files: [{
          expand: true,
          cwd: 'build/css',
          src: ['*.css', '!*.min.css'],
          dest: 'build/css',
          ext: '.min.css'
        }]
      }
    }


  });

  grunt.loadNpmTasks('grunt-contrib-cssmin');
  grunt.loadNpmTasks('grunt-contrib-less');

  grunt.registerTask('default', ['less', 'cssmin']);
};
```

Cal tenir en compte que tot el codi de la configuració es troba dintre de la funció exportada (module.exports = function (grunt)). Dins d'aquesta funció es poden distingir 3 seccions:

- **Inicialització** (grunt.initConfig): conté la configuració de les tasques (que corresponen als connectors). En aquest cas, less i cssmin. Aquí s'indiquen els fitxers d'origen i destí de cada connector, entre altres opcions.
- **Càrrega dels connectors** (grunt.loadNpmTasks): carrega els connectors.
- **Enregistrament de la tasca** (grunt.registerTask): s'enregistra la tasca amb el nom i la llista de tasques i l'ordre en què s'han d'executar.

Una vegada creat el fitxer Gruntile.js, per poder comprovar que heu realitzat les tasques correctament, heu de crear un directori anomenat *less* i dos fitxers anomenats full1.less i full2.less (si no els heu creat anteriorment).

Afegiu al fitxer full1.less el codi següent:

```
@mida-font: 15px;
@color: #ff0000;

p {
  font-size: @mida-font;
  color: @color;
}
I al fitxer full2.less, afegiu-hi el següent:
@color-principal: blue;
@color-vora: darken(@color-principal, 20%);
@color-enllac: lighten(@color-principal, 20%);

div {
  color: @color-principal;
  border: 1px solid @color-vora;

  a {
    color:@color-enllac;
  }
}
```

Arribats a aquest punt només cal executar l'ordre grunt a la línia d'ordres per comprovar que es realitza el processament correctament. La sortida serà similar a la següent:

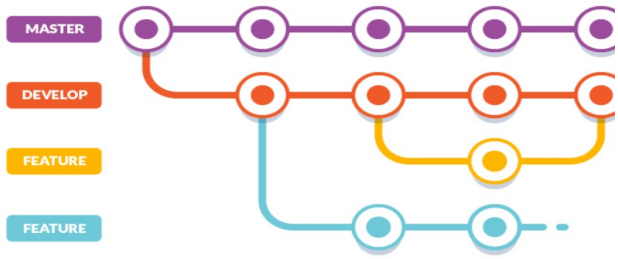
```
Running "less:dist" (less) task
>> 1 stylesheet created.

Running "cssmin:target" (cssmin) task
>> 1 file created. 124 B → 81 B

Done.
```

Una vegada finalitzat, s'haurà creat un directori anomenat *build/css* que contindrà dos fitxers: estils.css i estils.min.css, amb la versió minimitzada i el contingut següent:

```
p{font-size:15px;color:red}div{color:#00f;border:1px solid #009}div a{color:#66f}
```

Fixeu-vos que, al contrari que en el cas de Gulp, el processament de les tasques és seqüencial, és a dir, fins que no acaba la tasca less no s'inicia la tasca cssmin.

Tema 6: Control de versions i documentació



Índex:

1. Introducció

És molt habitual que el desenvolupament d'una aplicació no finalitze a l'hora d'entregar una primera versió al client, sinó que normalment s'ha de fer un manteniment i s'han d'afegir noves característiques a l'aplicació, especialment en l'àmbit del desenvolupament web, on les aplicacions desenvolupades acostumen a fer servir el model de programari com a servei (*software as a service* o SAAS, en anglès).

Això implica que l'aplicació ha d'estar ben documentada, ja que s'ha de continuar treballant en elles i pot ser que la implementació de noves funcionalitats ni tan sols la duga a terme la mateixa persona. Per altra banda, cal garantir que el desenvolupament de noves característiques i resolució d'errades no afecte la versió actual del programari, al mateix temps que pot ser necessari arreglar versions anteriors del programari.

En l'apartat 2 **"Documentació d'aplicacions"** aprendreu a documentar aplicacions utilitzant el format Javadoc, per documentar aplicacions en Java, i JSDoc, per documentar aplicacions en JavaScript, així com a exportar la documentació en format HTML per poder consultar-la externament i utilitzar eines col·laboratives per ampliar la documentació.

En l'apartat 3 **"Sistemes de control de versions"** aprendreu què són els sistemes de control de versions i com s'utilitzen. Detallarem la utilització de Git, les ordres més importants per gestionar el control de versions i com s'integra amb els entorns integrats de desenvolupament. Per acabar l'apartat, coneixereu GitHub i com crear repositoris remots per sincronitzar amb els repositoris locals.

2. Documentació d'aplicacions

Per desenvolupar un projecte informàtic cal generar molts tipus diferents de documentació a cada fase del projecte. Els elements que s'han de documentar depenen de la metodologia utilitzada per planificar el projecte. Per exemple, si s'utilitza una metodologia en cascada la documentació ha de ser exhaustiva, mentre que en el cas d'aplicar metodologies àgils el nombre d'elements és molt menor.

Entre els diferents tipus de documentació cal destacar la documentació del codi, ja que els encarregats de generar aquesta documentació són els programadors i aquesta s'inclou en forma de comentaris al codi font dels programes, encara que és possible extreure-la (en format HTML, per exemple) per consultar-la externament.

Per altra banda, com que habitualment en el desenvolupament d'un projecte informàtic hi participa més d'una persona, també és molt habitual trobar la documentació com a lloc wiki. A diferència de la documentació del codi, que és generat automàticament a partir dels comentaris, la documentació d'un lloc wiki està escrita pels diferents membres del projecte.

S'ha de diferenciar entre els formats de documentació i les eines per generar la documentació externa, ja que un mateix generador pot acceptar un o més formats diferents, però un format no pot barrejar-se amb els altres.

També cal destacar que la utilització d'un format o altre no depèn només del llenguatge, sinó que diferents biblioteques o entorns de treball poden fer servir formats diferents.

Per exemple, Dojo, un entorn de treball (*framework*) per a JavaScript, fa servir el seu propi format de documentació, com podeu veure a continuació:

```
// summary:
//        Representa un llibre.
// titol: Integer
//        Títol del llibre
// autor: String
//        Autor del llibre
function Llibre(titol, autor) {}
```

La mateixa informació documentada en format JSDoc, un altre format de documentació per a JavaScript, és:

```
/**
 * Representa un llibre.
 * @constructor
 * @param {number} titol - Títol del llibre
 * @param {string} autor - Autor del llibre
 */
function Llibre(titol, autor) {}
```

Tots dos formats són aplicats a una funció en el llenguatge JavaScript, però el format és completament diferent. Tingueu en compte que l'última paraula sobre el format que cal utilitzar la té el cap del projecte.

2.1. Avantatges de documentar el codi

La tasca de documentar el codi de l'aplicació fent servir comentaris no només serveix perquè puga ser consultat per tercers, sinó per a vosaltres mateixos, ja que al llarg de la vida d'un projecte cal fer canvis, millores i manteniment.

Per exemple, potser us heu d'encarregar d'arreglar una funció que implementareu fa més de sis mesos, i durant aquest temps segurament heu estat treballant en altres projectes. Si el codi està correctament documentat, no us costarà gens entendre com funciona, però, si no hi ha cap tipus de documentació, haureu de depurar i inspeccionar el codi per entendre de nou el seu funcionament. Un altre cas molt habitual és haver de treballar amb codi de tercers. Si aquest codi no està documentat, el temps que invertireu en entendre com funciona i com solucionar el problema i aplicar els canvis augmentarà enormement.

En el cas de treballar en projectes de programari lliure aquesta tasca és encara més important, ja que el codi s'utilitza i es modifica no només pel vostre equip de desenvolupament, sinó per qualsevol persona interessada. En aquests casos cal parar especial atenció a la documentació dels components públics de l'aplicació perquè són els elements als quals tindran accés altres desenvolupadors.

En qualsevol cas, cal tenir en compte que, igual que en el cas dels comentaris, una documentació incorrecta o desactualitzada és pitjor que una documentació inexistent, ja que els usuaris confien en el principi de veracitat.

2.1.1. Bones pràciques de programació

Cal tenir en compte que no es recomana utilitzar comentaris al codi quan no formen part de la documentació, perquè no són mantinguts i poden arribar a confondre altres desenvolupadors.

Molt pitjor que no trobar cap comentari respecte a un fragment de codi és trobar un comentari desfasat o que continga informació errònia. En molts casos aquests comentaris (que ningú recorda qui els va afegir ni quina funció tenen) es van arrossegant al llarg dels anys a mesura que el codi es modifica.

En comptes d'això, el que es recomana és utilitzar bones pràctiques per anomenar els elements del programari (classes, funcions, variables, etc.) i escriure el codi de manera comprensible.

Per exemple, en cas d'utilitzar noms de variables críptics, cal afegir comentaris per explicar què fa un fragment de codi. Per altra banda, la documentació del codi permet pal·liar algunes de les deficiències dels llenguatges o entorns de treballs, i això ajuda a comunicar la intencionalitat del codi a altres usuaris. Per exemple, a JavaScript no existeix el concepte d'àmbit privat, públic o protegit, però es pot etiquetar el codi indicant quin hauria de ser el seu àmbit:

```
// Conté el preu total d'una línia de comanda
// que consisteix en la multiplicació del preu
// unitari per la quantitat multiplicat pels
// impostos afegint el cost d'enviament.
float t = p * q * 0.21f + e;
```

En canvi, si es dona un nom clar a les variables, s'entén clarament què fa el codi:

```
const float IVA = 0.21f;
float totalLinia = preuUnitari * quantitat * IVA + enviament;
```

Com podeu veure, en el segon cas no cal cap comentari: es pot deduir correctament el funcionament i la intencionalitat del codi.

En el cas que el codi siga massa complicat i que tot i utilitzar noms afortunats no s'entenga, el que cal fer és reescriure el codi d'una manera més simple. Recordeu que això no només ajuda altres desenvolupadors que hagen de treballar amb el codi, sinó a vosaltres mateixos en el futur, si l'heu de mantenir.

2.1.2. Ressaltat de sintaxi als entorns de desenvolupament integrats (IDE)

A l'hora de desenvolupar una aplicació el més habitual és treballar amb un entorn de desenvolupament integrat (*integrated development environment*, IDE, en anglès). L'IDE concret amb el qual treballeu dependrà de les característiques del projecte o de l'equip de desenvolupament, ja que és possible que us siga imposat per l'empresa.

Alguns IDE molt coneguts són Eclipse, IntelliJ, Visual Studio, XCode o NetBeans.

Si el vostre codi està documentat utilitzant un codi reconegut pel vostre IDE, aquesta documentació pot mostrar-se dinàmicament ampliant la informació sobre els components de l'aplicació i millorar l'eficàcia del ressaltat de sintaxi i les funcions d'autocompletar.

La informació proporcionada pot incloure la descripció de funcions amb els seus paràmetres i valors de retorn, la informació sobre classes, els detalls sobre les variables, etc. Normalment per visualitzar-la només cal posar el ratolí sobre de l'element del qual voleu més informació.

Quant al ressaltat de sintaxi, cal destacar que l'IDE pot deduir els tipus de dades que accepta una funció com a paràmetre a partir de la documentació (a més del nombre de paràmetres esperats). D'aquesta manera pot ressaltar amb un senyal d'alerta els punts en els quals es detecten incongruències. Aquesta característica és especialment interessant en el cas de treballar amb llenguatges dèbilment tipats com JavaScript.

Per exemple, si en un programa en JavaScript escriviu una funció per manipular una cadena de text i posteriorment la invoqueu passant un número com a paràmetre, no rebeu cap error ni a l'IDE ni en executar la invocació, però el programa no funciona correctament, ja que el tipus de dada no és l'esperat.

En canvi, si heu documentat correctament la funció (i el vostre IDE inclou aquesta funcionalitat), en intentar passar un element d'un tipus diferent es ressalta el paràmetre de manera que és fàcil identificar que hi ha algun tipus de problema.

2.1.3. Indicacions per a compiladors: Closure Compiler

El Closure Compiler és una eina de Google per fer que el codi JavaScript es descarregue i s'execute més ràpidament. Com que el codi que executen els navegadors és JavaScript i no codi màquina, el que fa realment aquest compilador és analitzar el codi, eliminar el que no serveix i minimitzar-lo.

Com ja sabeu, JavaScript és un llenguatge dèbilment tipat i per aquesta raó el Closure Compiler requereix que la informació referent als tipus de les variables, paràmetres, retorn de funcions, etc, li siga proporcionat. Donat que és un compilador per a JavaScript, utilitza el format JSDoc (un format de documentació per a JavaScript) afegint algunes etiquetes extres per fer més estricte el llenguatge.

Entre les etiquetes que utilitza el Closure Compiler hi ha:

- **@const**: indica que una variable ha de tractar-se com una constant.
- **@constructor**: indica que la funció és un constructor.
- **@private**, **@protected**, @public: indiquen l'àmbit d'un mètode o propietat (concepte no aplicable a JavaScript).
- **@deprecated**: indica que la funció està obsoleta i desapareixerà en futures versions.
- **@extends**: indica que el constructor hereta d'una altra classe.
- **@implements**: indica que el constructor implementa una interfície.

- **@override**: indica que aquesta funció en sobreescriu una del mateix nom de la superclasse.

Les etiquetes s'utilitzen dins dels blocs de documentació, com podeu veure a continuació:

```
/**
 * Una forma
 * @interface
 */
function Forma() {};
Forma.prototype.dibuixa = function() {};

/**
 * Un triangle
 * @interface
 * @extends {Forma}
 */

function Triangle() {};
Triangle.prototype.obtenirCostats = function() {};
```

Aquest tipus d'etiquetes, en ser utilitzades conjuntament amb el Closure Compiler, permeten assegurar que es respecta la intencionalitat del codi. Per exemple, mostrant alertes si s'intenta canviar el valor d'una variable definida com a constant o si s'intenta accedir externament a un mètode etiquetat com a privat.

2.2. Documentació de Java: Javadoc

Javadoc és un generador de documentació per al llenguatge Java. S'utilitza des de la primera versió de Java i s'actualitza amb cada nova versió. Aquesta eina es troba inclosa en el Java Development Kit (JDK), de manera que si desenvolupeu aplicacions amb Java és molt probable que ja estiga instal·lada al sistema.

El format utilitzat per Javadoc és l'estàndard *de facto*. Per aquest motiu es troben molts generadors amb un format similar, com JSDoc i phpDocumentor. A més, es pot accedir al generador de documentació directament des de les IDE més populars, com són IntelliJ, Eclipse o NetBeans.

Cal tenir en compte que els comentaris són descartats quan es compila el codi font, de manera que documentar el codi no afecta el rendiment del programa.

2.2.1. Instal·lació del JDK i Javadoc

Javadoc es troba inclòs en la instal·lació del JDK.

Per a la instal·lació del JDK primer cal marcar la casella *Accept License Agreement* i seleccionar la versió adequada per al sistema operatiu. Un cop descarregat i instal·lat el JDK, cal comprovar que l'eina estiga instal·lada. Per això escriviu a la línia d'ordres:

```
javadoc
```

Tingueu en compte que el procés d'instal·lació no inclou la configuració de les variables d'entorn del sistema operatiu, i és molt possible que no funcione directament, ja que la variable PATH del vostre entorn ha d'incloure la ruta en la qual es troba.

En cas de no trobar-lo, heu de cercar el fitxer javadoc. La seva localització és diferent segons el sistema operatiu:

- Windows i Linux : dintre de la carpeta *bin* del directori d'instal·lació del JDK. Per exemple: C:/Arxius de programa/Java/jdk.1.8.0_131/bin/javadoc.exe.
- macOS: a */usr/bin/javadoc*, i segurament funcionarà desde qualsevol directori sense fer res.

Un cop localitzat, heu de modificar la variable PATH per afegir la ruta a la carpeta on es troba o copiar el fitxer javadoc a un directori que ja estiga configurat a la variable.

Cal destacar que per utilitzar l'eina des d'un IDE no cal canviar la configuració de l'entorn, ja que l'IDE té configurada correctament la ruta fins al JDK i totes les seues eines.

2.2.2. Format de Javadoc

El format de Javadoc consisteix a afegir un comentari multilínia immediatament abans del codi per documentar, com podeu veure en el següent exemple:

```
/**
 * Representa un llibre.
 */
public class Llibre {}
```

Alternativament, quan no hi ha moltes notes que afegir, es pot utilitzar el format curt. Per exemple, la classe Llibre es podria haver documentat de la següent manera:

```
/** Representa un llibre. */
public class Llibre {}
```

Javadoc permet utilitzar codi HTML incrustat als comentaris per millorar l'aspecte de la documentació generada, però no és recomanable utilitzar-lo, ja que empitjora la llegibilitat de la documentació directament sobre el codi. Per exemple, per afegir un text en negreta a la descripció de la classe Llibre es pot fer:

```
/** Representa un <b>llibre</b>. */
public class Llibre {}
```

Per afegir informació addicional a la documentació, Javadoc fa servir un sistema d'etiquetes prefixat amb el símbol de l'arrova. Per exemple: @author, @param o @return. Cal destacar que el nombre d'etiquetes de Javadoc és molt més reduït que el d'altres generadors com JSDoc, ja que, com que Java és un llenguatge fortament tipat, la majoria de les restriccions es troben definides pel mateix codi i no cal indicar-les. Per exemple: la privacitat de les propietats i mètodes, quan es tracta d'una classe i quan es tracta d'una funció.

A continuació podeu trobar una llista de les etiquetes més utilitzades a Javadoc:

- **@author** nom: indica l'autor del codi.
- **@deprecated** text: indica que aquest mètode o classe no s'ha d'utilitzar i, opcionalment, el motiu.
- **@exception** *classe descripció*, **@throw** *classe descripció*: són sinònimes i indiquen que aquest mètode pot llençar una excepció.

- **@param** *nom descripció*: afegeix informació sobre un paràmetre.
- **@return** *descripció*: afegeix informació sobre el valor de retorn d'un mètode.
- **@link** *paquet.classe#membre etiqueta*: insereix un enllaç que apunta a un altre element de la documentació.
- **@see** *referència*: indica que aquest element està relacionat amb un altre. Poden afegir-se múltiples etiquetes @see en un mateix comentari, cadascuna en una línia.
- **@since** *text*: indica en quina versió del programari es va afegir aquesta classe o mètode.
- **@version** *text*: indica la versió.

Molts entorns de treball i biblioteques de Java utilitzen anotacions en temps d'execució. Per exemple, Hibernate i Spring.

Cal destacar que en altres entorns s'utilitza el terme etiqueta i anotació indistintament, però en el cas de Javadoc la diferència és molt clara:

- **Etiqueta**: la intencionalitat és afegir estructura i contingut a la documentació, però mai afecta la semàntica del programa.
- **Anotació**: no afecta directament la semàntica del programa, però afecta la manera en què és tractat per eines i biblioteques. Les anotacions poden llegir-se en temps d'execució mitjançant la refracció.

2.2.3. Cas pràctic: documentació amb Javadoc i NetBeans

La documentació d'un programa amb Javadoc és molt més simple que en altres llenguatges gràcies a la rigidesa de Java. A continuació podeu veure una classe senzilla en Java que inclou alguns mètodes i propietats:

```
public class Persona {

    public String nom;
    public String cognom;

    private final String nomComplet;
    private float posicio;

    public Persona(String _nom, String _cognom) {
        nomComplet = nom + " " + cognom;
        nom = _nom;
        cognom = _cognom;
    }

    public void caminar(float _distancia) {
        posicio += moure(_distancia);
    }

    protected float moure(float _distancia) {
        return _distancia * 1.0f;
    }

    public void parlar(String _missatge) {
        System.out.println(generarMissatge(_missatge));
    }

    private String generarMissatge(String _missatge) {
```

```
            return nomComplet + " (" + posicio + "m): " + _missatge;
        }
    }
```

Com que la privacitat de les propietats i mètodes ve definida pel codi, així com el tipus dels paràmetres i els valors de retorn, la documentació d'aquests tipus de codi és molt senzilla. El mateix codi documentat en format Javadoc quedaria de la següent manera:

```
    /**
     * Representa una persona que pot caminar i parlar.
     *
     * @author Xavier Garcia
     * @version 1.0.0
     * @since 1.0.0
     */
    public class Persona {

        public String nom;
        public String cognom;

        private final String nomComplet;
        private float posicio;

        /**
         * El constructor s'encarrega de generar el nom complet de la persona.
         *
         * @param _nom      nom de la persona
         * @param _cognom   cognom de la persona
         */
        public Persona(String _nom, String _cognom) {
            nomComplet = nom + " " + cognom;
            nom = _nom;
            cognom = _cognom;
        }

        /**
         * Modifica la posició de la persona.
         *
         * @param _distancia    distància en metres
         */
        public void caminar(float _distancia) {
            posicio += moure(_distancia);
        }

        /**
         * Calcula la distància a moure, aquest mètode pot ser sobreescrit
         * per les subclasses.
         *
         * @param _distancia    distància en metres
         * @return              distància modificada
         */
        protected float moure(float _distancia) {
            return _distancia * 1.0f;
        }

        /**
         * Mostra un missatge per la pantalla.
         *
         * @param _missatge     missatge a mostrar
         */
        public void parlar(String _missatge) {
```

```
            System.out.println(generarMissatge(_missatge));
        }

        /**
         * Genera un missatge a partir de la cadena passada com argument i la
         * informació de la persona.
         *
         * @param _missatge missatge a incloure
         * @return          missatge generat
         */
        private String generarMissatge(String _missatge) {
            return nomComplet + " (" + posicio + "m): " + _missatge;
        }
    }
```

Com podeu apreciar, s'han fet servir tabulacions per separar els noms i les descripcions i per alinear les descripcions de retorn, però això és opcional. Només s'ha de tenir compte de mantenir el mateix criteri al llarg de tot el projecte.

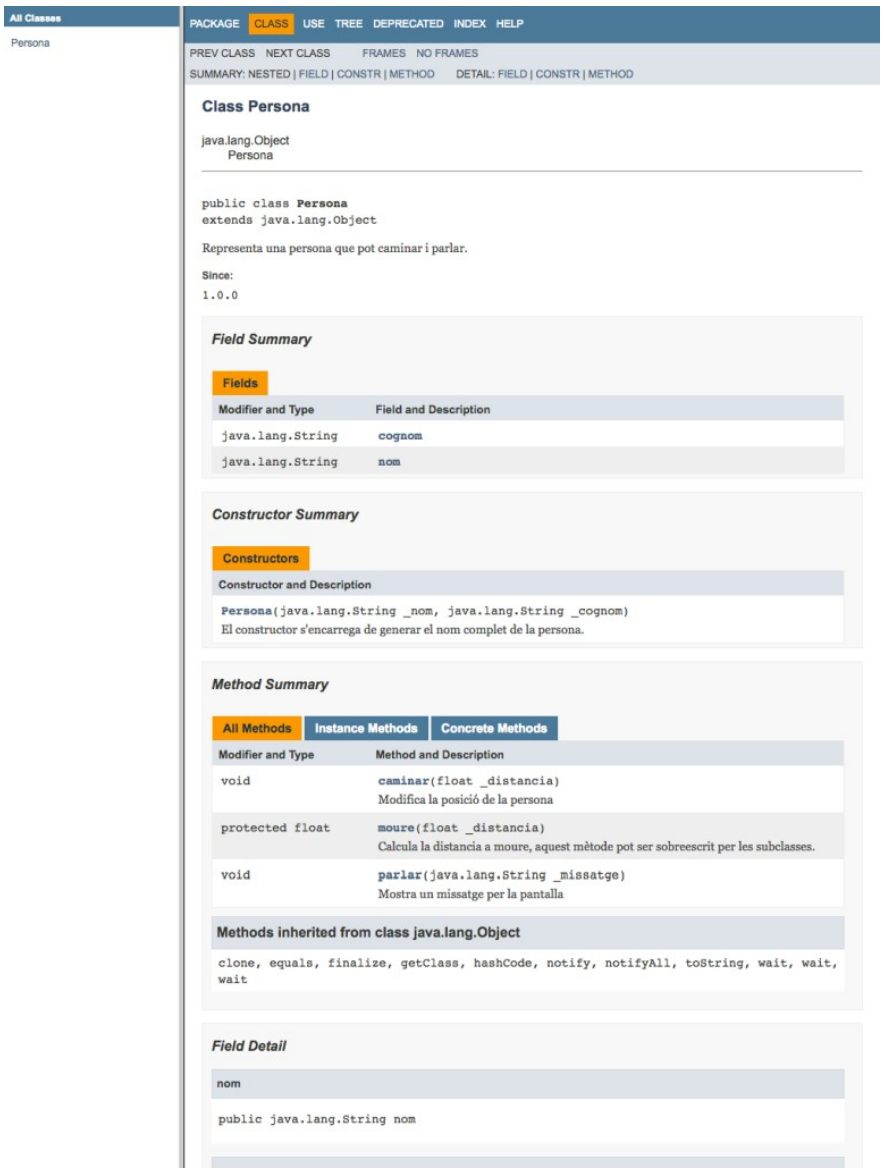
Un cop documentat el fitxer per generar la documentació en format HTML amb Netbeans només cal seguir els següents passos:

1. Obrir al NetBeans el projecte que inclou el fitxer amb el codi Java documentat.
2. Seleccionar l'opció ***Run> Generate Javadoc*** de la barra de menú.
3. Comprovar que a la finestra d'eixida de NetBeans es mostra el procés completat amb èxit.

L'eixida ha de ser similar a la següent:

```
ant -f /Users/xavier/NetBeansProjects/Documentacio -
Dnb.internal.action.name=javadoc javadoc
init:
Warning: Leaving out empty argument '-windowtitle'
Generating Javadoc
Javadoc execution
Loading source file
/Users/xavier/NetBeansProjects/Documentacio/src/Persona.java...
Constructing Javadoc information...
Standard Doclet version 1.8.0_131
Building tree for all the packages and classes...
Building index for all the packages and classes...
Building index for all classes...
Browsing:
file:/Users/xavier/NetBeansProjects/Documentacio/dist/javadoc/index.html
javadoc:
BUILD SUCCESSFUL (total time: 1 second)
```

En cas que es produeisca cap error, a l'eixida s'indica quin és. Cal solucionar-lo i tornar a generar la documentació. Un cop generada, s'obre automàticament una finestra al navegador que mostra la documentació en format HTML, com podeu veure a la figura



Cal tenir en compte que la documentació es crea dins del directori *dist/javadoc* del projecte, i no és visible des de la pestanya *Projects* de la zona de navegació. Heu de fer clic a la pestanya *Files* per poder visualitzar-lo des de NetBeans.

2.3. Documentació de JavaScript: JSDoc

Per al llenguatge JavaScript no hi ha cap eina estàndard per generar documentació, però un dels formats més estesos és JSDoc (i les seves variants). El seu format és molt similar al de Javadoc (per a Java) i phpDocumentor (per a PHP), així que l'adaptació és molt ràpida per a programadors que ja coneixen aquests altres sistemes.

Originalment JSDoc requeria la utilització del motor de navegador Rhino i la documentació era molt escassa, sense cap exemple pràctic, raó per la qual tenia molts detractors.

Afortunadament les versions més recents funcionen sobre Node.js; això fa que la seua instal·lació i utilització siga molt més simples. A més, es pot trobar documentació completa i amb exemples d'ús al seu lloc web: usejsdoc.org.

La seua utilització és molt simple: una vegada s'ha documentat el codi en aquest format, només cal executar el generador des de la línia d'ordres i es genera un lloc web amb tota la documentació del projecte. El generador permet modificar la configuració mitjançant diferents opcions des de la línia d'ordres, utilitzar un fitxer de configuració en format JSON i modificar les plantilles a partir de les quals es genera la documentació (incloent-hi els fitxers CSS i JS utilitzats).

2.3.1. Instal·lació de JSDoc

Per començar a utilitzar JSDoc no cal instal·lar cap programari, ja que només s'ha d'escriure la documentació directament al codi com a comentaris, però sí que és necessari instal·lar un programari específic per generar la documentació. La versió actual de JSDoc és una eina de Node.js i s'instal·la mitjançant el gestor de paquets npm.

Podeu trobar informació sobre Node.js i com instal·lar npm a l'apartat "Utilització de serveis de xarxa i automatització" de la unitat "Aplicacions web i serveis".

Per instal·lar-lo de forma global des de la línia d'ordres escriviu:

```
npm install jsdoc -g
```

Un cop instal·lat, per comprovar que s'ha instal·lat correctament podeu escriure jsdoc -v a la línia d'ordres. Us mostrarà un missatge amb una versió similar a:

```
JSDoc 3.4.3 (Thu, 10 Nov 2016 00:25:10 GMT)
```

Una vegada s'ha comprovat que tot funciona, per generar la documentació només cal indicar el nom del fitxer o el directori a partir del qual es vol generar la documentació:

```
jsdoc nom_directori
```

En executar l'ordre es crea un directori anomenat *out* que conté una sèrie de fitxers HTML i tres directoris: *fonts*, *scripts* i *styles*. Aquests directoris contenen les fonts, el codi JavaScript i els estils CSS utilitzats pel lloc web generat. Entre els fitxers HTML cal destacar index.html, que és la pàgina principal del lloc web i de la documentació.

2.3.2. Format de JSDoc

Per incloure documentació en format JSDoc a les aplicacions, cal incloure un comentari multilínia immediatament abans del codi per documentar, com podeu veure en el següent exemple:

```
/**
 * Representa un llibre.
 */
function Llibre() {}
```

Fixeu-vos que el format no és exactament igual que el format multilínia habitual:

- La documentació es tracta com un únic bloc, comença i acaba en la seva pròpia línia.

- La marca de començament inclou una barra i dos asteriscs en comptes d'un sol asterisc.
- Cada línia comença amb un espai seguit d'un asterisc i un altre espai abans d'afegir el text.
- L'última línia acaba amb un espai abans de la marca de tancament.

En cas que el codi per documentar sigui molt curt (per exemple, la declaració d'una propietat o variable) es pot utilitzar un format similar però en una sola línia, com podeu veure a continuació:

```
/** @type {string} */
var autor;
```

Una diferència important respecte a altres formats similars com Javadoc i phpDocumentor és que els tipus sempre s'escriuen entre claudàtors. Per exemple: {string}, {number} o {Cotxe}.

Igual que quan s'utilitza el format Javadoc, el text utilitzat per descriure els continguts pot incloure marques en HTML per enriquir la documentació (per exemple, les etiquetes per afegir un text en negreta o les etiquetes per afegir el text en cursiva), com podeu apreciar en el següent exemple:

```
/**
 * Representa un <i>llibre</i>.
 */
function Llibre() {}
```

Utilitzar aquesta característica no és gaire recomanable, ja que, si bé la documentació generada és més vistosa, la documentació incrustada al codi es fa més difícil d'entendre per als desenvolupadors que hi treballen directament.

Per afegir informació extra s'utilitzen etiquetes prefixades pel símbol de l'arrova (per exemple: @type, @param o @constructor). Aquestes etiquetes tenen un significat especial i són interpretades de manera diferent pel generador de documentació. En alguns casos aquestes etiquetes van acompanyades d'altres elements com poden ser un tipus i/o una descripció o nom.

Àmbit

Tot i que JavaScript no inclou moltes de les restriccions dels llenguatges clàssics com Java o C++, és possible indicar que s'han de respectar determinades restriccions mitjançant la documentació (tot i que només fent servir un compilador com el Closure Compiler és obligatori respectar-les).

Per aquesta raó, JSDoc inclou tot un seguit d'etiquetes per indicar aquestes restriccions:

- **@private**, **@protected**, **@public**: indiquen la privacitat de la classe, la propietat o el mètode.
- **@constant**, **@const**: indiquen que el codi representa una constant i el seu valor no pot canviar durant l'execució.
- **@global**: indica que el fragment de codi és global, és a dir, accessible des de qualsevol punt de l'aplicació, inclosa la consola de les eines de desenvolupador.

- **@readonly**: indica que una propietat és només de lectura.
- **@static**: indica que el fragment de codi és accessible directament des del constructor i no cal instanciar un objecte per accedir-hi.

Herència

JSDoc també permet documentar el codi de JavaScript com si es tractés d'un llenguatge clàssic, incloent-hi l'herència i l'ús d'interfícies i de classes abstractes, mitjançant les següents etiquetes:

- **@class**, **@constructor**: indiquen que la funció a continuació és un constructor i es pot invocar amb la paraula clau new per crear una instància de l'objecte.
- **@interface**: indica que el següent fragment de codi s'ha de tractar com una interfície.
- **@abstract**, **@virtual**: indiquen que la classe, objecte o mètode són abstractes i han de ser implementats pels descendents.
- **@augments** *nom_classe*, **@extends** *nom_classe*: indiquen que el següent fragment de codi és una classe que hereta de la classe indicada.
- **@implements** *nom_interficie*: indica que la classe implementa la interfície indicada.

Definició i utilització de tipus

Les etiquetes que permeten definir nous tipus i assignar un tipus a una variable o propietat són les següents:

- **@typedef** *{tipus} nom*: permet crear un nou tipus.
- **@property** *{tipus} nom*: indica una propietat del tipus definit. Aquesta etiqueta té altres usos, i aquest concretament no es troba correctament documentat a la pàgina oficial.
- **@type** *nom_tipus*: indica el tipus de la variable o propietat a continuació.

Un dels aspectes més complicats a l'hora d'utilitzar JSDoc és quan es vol concretar el tipus correcte d'un paràmetre o el valor de retorn d'una funció quan el tipus no és un dels tipus primitius com boolean, number o string.

En cas d'haver definit la classe, només cal indicar el seu nom entre claudàtors. Per exemple:

```
/**
 * Representa un llibre.
 * @class
 */
function Llibre() {}

/**
 * Enregistra un llibre.
 *
 * @params {Llibre} llibre - llibre a enregistrar
 */
function enregistrarLlibre(llibre) {
  // ... codi per enregistrar un llibre
}
```

En canvi, en altres casos pot ser que el tipus no s'hagi definit, s'hagi declarat literalment o no existeixi cap constructor. En aquest cas cal que definiu el vostre propi tipus mitjançant les etiquetes @typedef i @property. A continuació, podeu veure com es definiria el tipus Alumne:

```
/**
 * @typedef {Object} Alumne
 * @property {string} nom - com es diu aquest alumne
 * @property {number} edat - quants anys té aquest alumne
 */
```

Un cop definit, es pot utilitzar com qualsevol altre tipus. Vegeu-ho en el següent exemple:

```
/**
 * Matricula un alumne.
 *
 * @params {Alumne} alumne - alumne a enregistrar
 */
function matricularAlumne(alumne) {
  // ... codi per matricular un alumne
}
```

Hi ha altres sistemes per aconseguir el mateix resultat. Aquí només se n'ha mostrat un, el més simple, però si consulteu la documentació oficial de JSDoc o cerqueu per internet podeu trobar mètodes alternatius i exemples molt més complexos.

Un altre detall que cal tenir en compte és que JSDoc no discrimina si el nom del tipus es troba en majúscules o minúscules. Podeu trobar exemples en què es faci referència als tipus numèrics com number o com Number. En aquesta unitat s'ha decidit utilitzar el nom del tipus en minúscula quan es tracti de tipus predefinits per JavaScript, i el nom amb la inicial majúscula per als objectes propis, *Arrays* i el tipus objecte (Object).

Funcions i mètodes

Les etiquetes més utilitzades habitualment són les relatives a funcions i mètodes, ja que és important documentar què són els paràmetres que es passen a una funció i quin és el tipus del resultat. Per documentar aquesta informació, JSDoc facilita les següents etiquetes:

- **@function**, **@method**: indiquen que es tracta d'una funció. Donat que a JavaScript les funcions es poden passar com arguments i retornar des d'altres funcions, en alguns casos cal especificar que una variable o propietat referencia una funció, ja que no sempre pot ser determinat automàticament. També es pot utilitzar per etiquetar un mètode a un tipus definit.
- **@param** *{tipus} nom - descripció*: indica quin és el tipus del paràmetre amb el nom indicat i afegeix una descripció. El nom és obligatori, però el tipus i la descripció es poden ignorar, però no es pot incloure una descripció si no s'ha afegit també el tipus.

Documentació de paràmetres amb JSDoc

JSDoc inclou una gran quantitat d'opcions per permetre definir al més exactament possible els paràmetres de les funcions, incloent-hi paràmetres opcionals, tipus alternatius i utilització d'objectes complexos.

- **@returns** *tipus - descripció*: indica el tipus de retorn d'una funció o mètode, i opcionalment hi afegeix una descripció. Igual que en el cas dels paràmetres, aquesta etiqueta accepta que valors opcionals pel tipus (separats pel símbol de la canonada |). Vegeu un exemple de documentació d'una funció amb paràmetres i retorn:

```
/**
 * Divideix a entre b.
 *
 * @param {number} a - dividend
 * @param {number} b - divisor
 * @returns {string|number} - resultat
 */
function dividir(a, b) {
  if (b === 0) {
    return 'Error. Divisió per 0';
  }
  return a/b;
}
```

- **@override**: indica que aquesta funció sobreescriurà una altra funció amb el mateix nom de la superclasse.
- **@throws** *{tipus} descripció*: indica que aquesta funció pot llençar una excepció del tipus indicat si s'especifica. Tant el tipus com la descripció són opcionals.
- **@deprecated**: indica que aquest fragment de codi (habitualment una funció o mètode) està obsolet i no s'ha d'utilitzar, ja que en versions posteriors deixa d'estar disponible.
- **@requires**: indica que el bloc de codi requereix que estiguin carregats un mòdul o biblioteca concrets per funcionar.

'Events'

Events o esdeveniments?

Encara que la traducció al català seria esdeveniment, ens hi referim amb la nomenclatura en anglès perquè no hi hagi confusions.

Com que la gestió d'*events* és un factor fonamental a JavaScript, JSDoc també inclou les següents etiquetes per documentar-los:

- **@emits** *nom_classe#nom_event*, **@fires** *nom_classe#nom_event*: són sinònimes i indiquen que el fragment de codi pot disparar un *event*.
- **@event** *nom_classe#nom_event*: indica l'*event* que es dispara. Per exemple:

```
/**
 * Envia una notificació.
 *
 * @emits GestorNotificacions#notificacio_enviada
 */
GestorNotificacions.prototype.enviar = function(notificacio) {
  // ... codi per enviar la notificació

  /**
   * 'Event' notificacio_enviada
   *
```

```
     * @event GestorNotificacions#notificacio_enviada
     */
    this.dispatchEvent('notificacio_enviada', {
      // ... dades afegides a l'event
    });
}
```

Fixeu-vos que l'*event* es declara en el moment que s'utilitza, és a dir, s'està declarant notificacio_enviada i no pas la invocació al mètode emit. Per altra banda, per indicar que un fragment de codi detecta un *event* concret s'utilitza la següent etiqueta:

- **@listens** *nom_event*: indica que aquest fragment de codi detecta l'*event* especificat.

Informació addicional sobre el codi

Les següents etiquetes s'utilitzen per proporcionar informació addicional sobre el codi. Normalment acostumen a trobar-se a la capçalera del fitxer que conté el codi.

- **@author** *nom <adreça_de_correu>*: indica l'autor de la classe, el mòdul o el fragment de codi. Opcionalment es pot indicar l'adreça de correu electrònic, entre xebrons. Per exemple:

```
/**
 * Representa un <b>llibre</b>.
 *
 * @author Xavier Garcia <email@exemple.com>
 */
function Llibre() {}
```

- **@summary**: conté una descripció curta.
- **@version** *versió*: indica la versió d'un element (per exemple, @version 1.2.3).
- **@copyright** *text_legal*: indica la informació sobre els drets d'autor.
- **@license** *identificador*: indica la llicència que s'aplica al codi (GNU GPLv3, MIT License, Apache License 2.0, etc.).

2.3.3. Opcions de configuració del generador de documentació

Un cop documentat el codi amb el format JSDoc, es pot generar la documentació mitjançant la línia d'ordres indicant el nom del directori o del fitxer a partir del qual es vol generar la documentació:

```
jsdoc nom_directori -d directori_eixida
```

Especificant l'opció -d i un nom de directori, la documentació es genera en el directori indicat. En cas contrari, es genera al directori *out*.

Per defecte, la documentació generada no inclou els elements amb l'etiqueta @private, ja que es considera que aquests no formen part de l'API de l'aplicació. Si es volen mostrar, s'ha d'afegir l'opció -p o –private a la línia d'ordres, com podeu veure a continuació:

```
jsdoc nom_directori -p
```

En cas de voler realitzar accions més complexes, el més recomanable és utilitzar un fitxer de configuració, ja que permet un major nombre d'opcions i només cal configurar-lo un cop. A partir d'aquest moment s'aplica sempre que utilitzem l'opció -c. Per exemple, si el fitxer de configuració s'anomena conf.json, per utilitzar-lo heu d'escriure a la línia d'ordres:

jsdoc nom_directori -c conf.json

Configuració de JSDoc

El format d'aquest fitxer (que heu de crear vosaltres) ha de ser similar al següent:

```
{
    "tags": {
        "allowUnknownTags": true,
        "dictionaries": ["jsdoc","closure"]
    },
    "source": {
        "includePattern": ".+\.js(doc|x)?$",
        "excludePattern": "(^|\/|\\)_"
    },
    "plugins": [],
    "templates": {
        "cleverLinks": false,
        "monospaceLinks": false
    }
}
```

El significat d'aquest fitxer de configuració és el següent:

- JSDoc permet la utilització d'etiquetes desconegudes (tags.allowUnknownTags).
- S'admeten tant les etiquetes de JSDoc com les del Closure Compiler (tags.dictionaries).
- Només es processaran els fitxers acabats en: ".js", ".jsdoc" i ".jsx" (source.includePattern).
- Qualsevol fitxer o directori que comenci per una barra baixa serà ignorat (source.excludePattern).
- No es carrega cap connector (plugins).
- L'etiqueta @link es representarà com a text pla (templates.cleverLinks, templates.monospaceLinks).

En aquest fitxer es poden incloure també algunes de les opcions acceptades per la línia d'ordres mitjançant la propietat opts com es veu en el següent exemple:

```
{
  "opts": {
    "template": "templates/default",  // És el mateix que -t
templates/default
    "destination": "./sortida/",      // És el mateix que -d ./sortida/
          "private":
true                                                  // És el mateix que -p
  }
}
```

En aquest cas s'utilitzarà la plantilla per defecte, ja que és la que es troba a la ruta indicada, el directori de destí s'anomenarà *sortida* i es mostraran els elements privats.

Configurar la plantilla per defecte

Fixeu-vos que a l'opció template, el directori sobre el qual es treballa és el d'instal·lació de JSDoc i no l'actual. En canvi, a l'opció destination sí que ho és, i, per tant, el directori es crea al mateix directori des del qual s'hagi executat l'ordre.

JSDoc permet utilitzar diferents plantilles per generar la documentació, però només n'inclou tres:

- **default**: és la predeterminada i genera la documentació en format HTML.
- **haruki**: és una plantilla experimental i genera la documentació en format JSON o XML per ser utilitzada per altres aplicacions.
- **silent**: no genera cap resultat, serveix per validar la correcció del codi, ja que si el processament no és correcte es visualitzen els errors per la consola a l'executar el JSDoc.

Podeu veure aquestes plantilles al repositori de JSDoc al següent enllaç: goo.gl/g5kMg2. En aquest mateix enllaç hi ha també les instruccions per crear la vostra pròpia plantilla i com carregar-la (heu de donar també una ullada al fitxer publish.js de la plantilla *default*). Fixeu-vos que la ruta a la vostra plantilla ha de ser absoluta, mentre que quan s'utilitza una de les plantilles predefinides la ruta és relativa al directori d'instal·lació de JSDoc.

Crear una nova plantilla no és una tasca trivial, a més de ser innecessari en la majoria dels casos. Generalment és suficient modificant la composició. Per fer-ho, podeu indicar al fitxer de configuració que carregui el vostre propi fitxer de composició.

El fitxer que conté la composició de la plantilla per defecte s'anomena layout.tmpl i el podeu trobar a *jsdoc/templates/default/tmpl/layout.tmpl*. Per crear el vostre propi *layout* podeu copiar-lo en el vostre directori i modificar-lo.

En aquest cas s'ha conservat el nom del fitxer, però no és necessari. Podeu utilitzar el nom que vulgueu.

Una vegada esteu satisfets amb els canvis, podeu carregar-lo en lloc de la composició per defecte, afegint dins del fitxer de configuració el següent codi:

```
{
  "templates": {
    "default": {
        "layoutFile": "./layout.tmpl"
        }
  }
}
```

Fixeu-vos que aquest fitxer inclou la capçalera de la pàgina i el peu, incloent-hi la càrrega dels fitxers CSS i JavaScript que s'utilitzaran. Només heu d'afegir els vostres fulls d'estil per adaptar-la a les vostres necessitats. Cal destacar que la ruta és relativa al directori on s'ha executat JSDoc i no al directori d'instal·lació.

Només queda un petit detall per poder donar per finalitzada la configuració de JSDoc. Com heu vist quan es genera la documentació, es creen múltiples carpetes amb diferents fitxers automàticament: codi JavaScript, codi CSS i fonts. És possible que necessiteu

incloure els vostres propis fitxers estàtics: per exemple, els vostres fulls d'estil, fitxers de codi o imatges. Per afegir-los només heu d'incloure la llista de fitxers o directoris dintre de l'opció de configuració templates, com en l'exemple següent:

```
{
  "templates": {
    "default": {
      "staticFiles": {
        "include": [
            "./fitxers_estatics"
        ]
      }
    }
  }
}
```

Una vegada afegida aquesta opció al fitxer de configuració, en generar la documentació tot el contingut del directori *fitxers_statics* s'afegeix al directori arrel de sortida, inclosos els directoris (excepte si són buits).

A continuació podeu veure com queda el fitxer de configuració amb els canvis a la plantilla i a les opcions de sortida:

```
{
        "opts": {
    "template": "templates/default",
          "destination": "./sortida/",
          "private": true
        },
        "templates": {
                "default": {
                        "layoutFile": "./layout.tmpl",
                        "staticFiles": {
                                "include": [
                                        "./fitxers_estatics"
                                ]
                        }
                }
        }
}
```

Cal destacar que tant la substitució del fitxer de composició com la inclusió dels fitxers estàtics s'han d'incloure dins de *default*, ja que són modificacions sobre la plantilla *default*.

2.3.4. Cas pràctic: Documentació amb JSDoc

Descàrrega de l'aplicació client-servidor-xat. Podeu descarregar-vos l'aplicació al següent enllaç: https://github.com/XavierGaro/client-servidor-xat/tree/4.0

Vegem un exemple pràctic de documentació de codi. Partint de la versió 3.0 (aquesta és la branca del repositori on es troba) d'una aplicació client-servidor de xat, es documenta el codi del client (JavaScript) que es troba al fitxer client-xat.js. Podeu trobar els fitxers originals al següent enllaç: https://github.com/XavierGaro/client-servidor-xat/tree/3.0.

El codi del client és el següent (*client-servidor-xat/Client/js/client-xat.js*):

```
var AplicacioXat = function () {
```

```
    var socol = io();

    $('form').submit(function(){
        socol.emit('missatge_xat', $('#missatge').val());
        $('#missatge').val('');
        return false;
    });

    socol.on('missatge_xat', function(msg){
        afegirMissatge(msg);
    });

    socol.on('missatge_estat', function(msg){
        afegirMissatge(msg, true);
    });

    afegirMissatge = function (msg, esEstat) {
        var className = esEstat ? "estat" : "";
        $('#missatges').append($('<li class='+ className + '>').text(msg));
    };
};
```

El primer que heu de fer és afegir la informació sobre el contingut del fitxer a la capçalera, indicant qui n'és l'autor, la versió, la descripció del que hi ha en el fitxer, etc.:

```
/**
 * Aquest fitxer inclou tota la funcionalitat necessària per connectar amb
 * un servidor de xat mitjançant WebSockets, i més concretament amb la
 * biblioteca Socket.io.
 *
 * @author Xavier Garcia <email@example.com>
 * @version: 3.0.1
 * @license GNU GPLv3
 * @summary Client simple per a una aplicació de xat mitjançant Socket.io
 */
```

A continuació es troba el constructor AplicacioXat. Aquesta funció pot etiquetar-se com a @constructor o com a @class (més proper als llenguatges clàssics). Utilitzar una etiqueta o altra és indiferent (són sinònimes), però cal utilitzar el mateix criteri al llarg de tot el projecte i no barrejar-les. A continuació podeu veure com es documenta aquesta classe:

```
/**
 * Constructor per instanciar aplicacions de xat que connecten amb un
servidor
 * remot, sincronitzen l'entrada de dades amb l'enviament al servidor i
 * mostren per pantalla les dades rebudes.
 *
 * @constructor
 * @requires jQuery
 * @requires Socket.io
 */
```

S'ha indicat que aquesta classe requereix les biblioteques jQuery i Socket.io, ja que sense no pot funcionar. Opcionalment es poden haver afegit a la capçalera del programa, però com que només hi ha una classe que els utilitzi és indiferent.

Tot i que a la documentació de JSDoc s'especifica que l'etiqueta @require s'ha d'utilitzar amb mòduls, és habitual utilitzar-lo també per a biblioteques. Recordeu que en última instància la documentació és una eina i podeu adaptar-la a les vostres necessitats.

Seguidament, dins del constructor AplicacioXat hi ha la declaració de la variable socol, a la qual s'assigna el retorn de la funció io. Consultant la documentació de la biblioteca Socket.io podeu veure que aquesta funció retorna un objecte de tipus Socket.

Com que el tipus Socket no és un tipus reconegut per JavaScript i la documentació de la biblioteca no forma part del vostre projecte, hi ha dues opcions:

- Utilitzar Socket com a tipus sense aportar cap altra informació. En aquest cas alguns IDE mostren el ressaltat d'alerta, ja que no reconeixen el tipus però no afecta en res al programa.
- Definir el tipus Socket afegint les propietats que s'utilitzen en aquest codi.

Els tipus definits amb @typedef i les funcions etiquetades com a @callback no es mostren correctament en generar la documentació HTML (última comprovació: versió 3.4.3).

La definició mínima del tipus Socket seria la següent:

```
/**
 * Sòcol de connexió a un servidor
 *
 * @typedef {Object} Socket
 * @method Socket.on - detecta quan es produeix un event.
 * @method Socket.emit - dispara un event.
 */
```

Un cop definit el tipus, es pot utilitzar com es veu al codi següent:

```
/**
 * @type {Socket}
 * @private
 */
var socol = io();
```

Fixeu-vos que en utilitzar dues etiquetes no és possible utilitzar el format de línia única, cada etiqueta s'ha d'escriure a la seva pròpia línia. S'ha etiquetat com a mètode privat, ja que no és possible accedir a aquesta variable des de fora de la classe.

El següent fragment de codi pot donar més mals de cap del que pot semblar a primera vista:

```
$('form').submit(function(){
  socol.emit('missatge_xat', $('#missatge').val());
  $('#missatge').val('');
  return false;
});
```

Per una banda, el codi detecta quan es dispara l'*event* submit al formulari i invoca al mètode emit de l'objecte socol. Això vol dir que l'encarregat de disparar l'*event* missatge_xat és l'objecte de tipus Socket, i no pas la classe AplicacioXat. Per consegüent, s'ha de modificar la definició de Socket per incloure els *events* que dispara i els que escolta. Per altra banda, s'han de definir els *events* que formen part de l'aplicació, que

són: missatge_xat i missatge_estat. Com que la definició del tipus Socket en depèn, es recomana afegir-los abans. La documentació dels *events* esmentats seria la següent:

```
/**
 * Event que indica que s'ha enviat o rebut un missatge de xat.
 *
 * @event AplicacioXat~missatge_xat
 */

/**
 * Event que indica que s'ha modificat l'estat.
 *
 * @event AplicacioXat~missatge_estat
 */
```

Un cop definits els *events* es poden fer els canvis necessaris a la definició del tipus Socket:

```
/**
 * Sòcol de connexió a un servidor
 *
 * @typedef {Object} Socket
 * @property {string} id - identificador del sòcol
 * @method on - detecta quan es produeix un event.
 * @method emit - dispara un event.
 * @emits AplicacioXat#event:missatge_xat
 * @listens AplicacioXat#event:missatge_xat
 * @listens AplicacioXat#event:missatge_estat
 */
```

Una vegada resolta la problemàtica de la detecció d'*events*, es pot procedir a documentar el codi. Com que $('form').submit no és un element documentable en el seu estat actual, per poder documentar-lo heu d'assignar el valor de ('form') a una variable (o propietat privada), de manera que aquesta sí que es podrà documentar:

```
var $form = $('form');

$form.submit(function(){
  socol.emit('missatge_xat', $('#missatge').val());
  $('#missatge').val('');
  return false;
});
```

Un cop creada, la propietat privada pot documentar-se. Fixeu-vos que es tracta d'un objecte jQuery i, per tant, cal definir el tipus com s'ha fet amb el sòcol, però en aquest cas només s'indica que és un objecte jQuery:

```
/**
 * @type {jQuery}
 * @listens submit
 * @private
 */
var $form = $('form');
```

Cal tenir en compte que l'*event* submit no es mostra a la documentació, ja que no s'ha documentat com a *event*. A continuació, pot documentar-se la funció que es passa com a paràmetre de la manera següent:

```
/**
```

```
     * Envia el contingut del quadre de text al servidor mitjançant
     * el sòcol i neteja el contingut del quadre de text.
     *
     * @callback
     * @returns {boolean}
     */
    function(){
      socol.emit('missatge_xat', $('#missatge').val());
      $('#missatge').val('');
      return false;
    }
```

Fixeu-vos que s'ha utilitzat l'etiqueta @callback per aclarir que aquesta funció serà invocada automàticament.

Ara ja podeu generar la documentació del fitxer amb l'ordre:

```
jsdoc client-xat.js -p
```

Si obriu el fitxer index.html al vostre navegador, veureu a la banda dreta la llista de classes (que només inclou AplicacioXat). Si feu clic a sobre, podreu veure la documentació de la classe.

En primer lloc veureu la descripció, seguida pels mòduls que requereix (jQuery i Socket.io). A continuació, les propietats: $form i Socket. Fixeu-vos que a continuació del nom de la propietat, separada per dos punts, se n'indica el tipus.

En el cas de $form el tipus és jQuery, però es veu ombrejat. Això és degut al fet que el tipus és desconegut pel generador de documentació. Per altra banda, al costat de socol es veu el tipus Socket en color blau, i es pot clicar. La diferència radica en el fet que el segon ha estat definit amb @typedef.

Malauradament, si feu clic sobre l'enllaç mostra una pàgina d'error indicant que no s'ha trobat el fitxer. Això és degut al fet que el generador de documentació no acaba de funcionar correctament amb els tipus definits (tot i que en alguns casos sí que funciona).

Un altre detall que cal tenir en compte és que no es veu per enlloc ni la documentació referent a la funció etiquetada com a @callback ni que la propietat $form detecta l'*event* submit. Aquesta informació forma part de la implementació interna de l'aplicació i no és exportada pel generador, només serà visible pels desenvolupadors, ja que es troba incrustada al codi.

És a dir, si documenteu les funcions disparades en detectar-se els *events* missatge_xat i missatge_estat, tampoc no es veuran a la documentació, però seran útils per a altres desenvolupadors.

A continuació vegeu com quedaria el codi d'aquestes funcions:

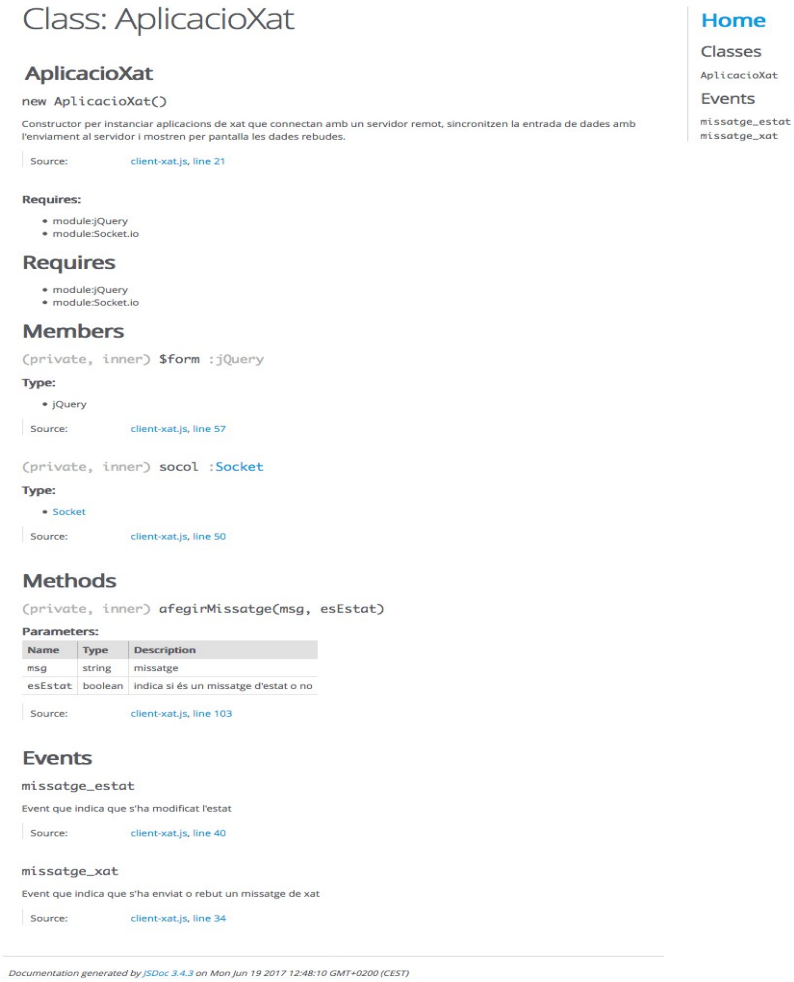
```
socol.on('missatge_xat',
  /**
   * Afegeix un missatge a la llista de missatges.
   *
   * @callback
   * @param {string} msg - missatge d'entrada
   */
```

```
  function(msg){
    afegirMissatge(msg);
  }
);

socol.on('missatge_estat',
/**
 * Afegeix un missatge de canvi d'estat la llista de missatges.
 *
 * @callback
 * @param {string} msg - missatge d'estat
 */
function(msg){
  afegirMissatge(msg, true);
  }
);
```

Finalment, a l'hora de documentar la funció afegirMissatge es considera un mètode privat i, per conseqüent, sí que serà visible a la documentació generada. La documentació d'aquest mètode seria:

```
/**
  * Afegeix un missatge al llistat de missatges.
  *
  * @private
  * @param {string} msg - missatge
  * @param {boolean} esEstat - indica si és un missatge d'estat o no.
  */
var afegirMissatge = function (msg, esEstat) {
  var className = esEstat ? "estat" : "";
  $('#missatges').append($('<li class='+ className + '>').text(msg));
};
```

Un cop es genera la documentació, el resultat final per a la documentació de la classe AplicacioXat és la que es pot veure a la figura:



Class: AplicacioXat

AplicacioXat

new AplicacioXat()

Constructor per instanciar aplicacions de xat que connectan amb un servidor remot, sincronitzen la entrada de dades amb l'enviament al servidor i mostren per pantalla les dades rebudes.

Source: client-xat.js, line 21

Requires:

- module:jQuery
- module:Socket.io

Requires

- module:jQuery
- module:Socket.io

Members

(private, inner) $form :jQuery

Type:

- jQuery

Source: client-xat.js, line 57

(private, inner) socol :Socket

Type:

- Socket

Source: client-xat.js, line 50

Methods

(private, inner) afegirMissatge(msg, esEstat)

Parameters:

Name	Type	Description
msg	string	missatge
esEstat	boolean	indica si és un missatge d'estat o no

Source: client-xat.js, line 103

Events

missatge_estat

Event que indica que s'ha modificat l'estat

Source: client-xat.js, line 40

missatge_xat

Event que indica que s'ha enviat o rebut un missatge de xat

Source: client-xat.js, line 34

Home

Classes

AplicacioXat

Events

missatge_estat

missatge_xat

Documentation generated by JSDoc 3.4.3 on Mon Jun 19 2017 12:48:10 GMT+0200 (CEST)

El codi complet documentat quedaria de la següent manera:

```
/**
 * Aquest fitxer inclou tota la funcionalitat necessària per connectar amb
 * un servidor de xat mitjançant WebSockets, i més concretament amb la
 * biblioteca Socket.io.
 *
 * @author Xavier Garcia <email@example.com>
 * @version: 3.0.1
 * @license GNU GPLv3
 * @summary Client simple per una aplicació de xat mitjançant Socket.io
 */

/**
 * Constructor per instanciar aplicacions de xat que connecten amb un
servidor remot, sincronitzen l'entrada
 * de dades amb l'enviament al servidor i mostren per pantalla les dades
rebudes.
 *
 * @constructor
 * @requires jQuery
 * @requires Socket.io
 */
var AplicacioXat = function () {

    /**
     * Sòcol de connexió a un servidor
```

```
 *
 * @typedef {Object} Socket
 * @method Socket.on - detecta quan es produeix un event.
 * @method Socket.emit - dispara un event.
 * @emits AplicacioXat#event:missatge_xat
 * @listens AplicacioXat#event:missatge_xat
 * @listens AplicacioXat#event:missatge_estat
 */

/**
 * Event que indica que s'ha enviat o rebut un missatge de xat.
 *
 * @event AplicacioXat~missatge_xat
 */

/**
 * Event que indica que s'ha modificat l'estat.
 *
 * @event AplicacioXat~missatge_estat
 */

/**
 * @type {Socket}
 * @private
 */
var socol = io();

/**
 * @type {jQuery}
 * @listens submit
 * @private
 */
var $form = $('form');

$form.submit(
    /**
     * Envia el contingut del quadre de text al servidor mitjançant
     * el sòcol i neteja el contingut del quadre de text.
     *
     * @callback
     * @returns {boolean}
     */
    function () {
        socol.emit('missatge_xat', $('#missatge').val());
        $('#missatge').val('');
        return false;
    }
);

socol.on('missatge_xat',
    /**
     * Afegeix un missatge a la llista de missatges.
     *
     * @callback
     * @param {string} msg - missatge d'entrada
     */
    function (msg) {
        afegirMissatge(msg);
    }
);
```

```
        socol.on('missatge_estat',

            /**
             * Afegeix un missatge de canvi d'estat a la llista de missatges.
             *
             * @callback
             * @param {string} msg - missatge d'estat
             */
            function (msg) {
                afegirMissatge(msg, true);
            }
        );

        /**
         * Afegeix un missatge a la llista de missatges.
         *
         * @private
         * @param {string} msg - missatge
         * @param {boolean} esEstat - indica si és un missatge d'estat o no.
         */
        var afegirMissatge = function (msg, esEstat) {
            var className = esEstat ? "estat" : "";
            $('#missatges').append($('<li class=' + className +
            '>').text(msg));
        };

    };
```

Cal tenir en compte que per aprofitar la capacitat dels IDE per interpretar la documentació i ressaltar la sintaxi tots els tipus han d'estar perfectament definits i que les biblioteques i entorns de treball s'han d'afegir (a l'IDE) com a biblioteques externes.

Com podeu imaginar, afegir els tipus no definits i documentar els fragments de codi que no formen part de la documentació representa un esforç addicional que no aporta cap valor de cara a la generació de la documentació, tot i que pot ser útil per vosaltres mateixos i altres desenvolupadors que hagin de treballar amb el codi.

Per aquests motius, s'ha de valorar cas per cas si val la pena o no fer aquest esforç extra, ja que en molts casos el codi pot ser suficientment expressiu o simple i documentar-lo no té sentit, com podeu comprovar en el següent exemple:

```
/**
 * Suma dos números passats per paràmetre.
 *
 * @param {number} a - primer número a sumar
 * @param {number} b - segon número a sumar
 * @returns {number} - suma dels dos números passats per paràmetre
 */
function sum(a, b) {
  return a + b;
}
```

Per acabar, recordeu que la documentació és una eina més a la vostra disposició i l'heu d'utilitzar de la manera que millor s'adapte als vostres projectes. Potser hi ha casos en què heu de ser molt rigorosos, perquè la documentació serà utilitzada per altres programes de generació de codi (com el Closure Compiler) o aquesta s'exportarà en format HTML i

serà consultada per tercers, però en altres casos només cal documentar el mínim necessari, ja que la intencionalitat del codi serà clara (com a l'exemple anterior).

2.3.5. Variacions

Com que JSDoc és un format força estès, s'han generat algunes variacions o ampliacions.

Per exemple, el Closure Compiler utilitza aquest format per obtenir informació sobre l'estructura del codi que no és proporcionada pel llenguatge, com ara la privacitat dels mètodes i els tipus de dades. Per altra banda, el Closure Compiler afegeix noves etiquetes que són exclusives del compilador, com @nocollapse i @preserve.

Una altra variació és la utilitzada per AngularJS, un entorn de treball de JavaScript, anomenada ngdoc. Aquesta variant admet moltes de les etiquetes de JSDoc, especialment:

- **@name** *nom*: indica el nom del document ngdoc.
- **@param** *{tipus} nom descripció*: descriu un paràmetre d'una funció.
- **@returns** *{tipus} descripció*: descriu el que retorna una funció.
- **@requires**: indica que depèn d'altres serveis.
- **@property**: descriu una propietat d'un objecte.
- **@description**: és usada per afegir la descripció d'un component.
- **@link**: indica un enllaç a una URL o tipus de l'API.
- **@example**: indica un exemple formatat com un bloc de codi.
- **@deprecated**: indica que el codi següent està obsolet i no s'ha d'utilitzar.
- **@this**: especifica a què fa referència this en el context de la documentació.

A més, inclou altres directives pròpies que només són aplicables a les aplicacions desenvolupades amb AngularJS, com @ngdoc, @scope, @priority, @animations i @restrict.

Per generar la documentació, cal instal·lar un nou mòdul anomenat Dgeni. Aquest mòdul es pot instal·lar mitjançant el gestor de paquets npm.

Cal destacar que la configuració de Dgeni és més complexa que la de JSDoc, ja que per poder generar la documentació cal crear un paquet que consisteix en el programa de Node.js, amb la configuració i les dependències necessàries del projecte. Alternativament, podeu utilitzar-lo juntament amb un sistema d'automatització com Grunt o Gulp, ja que tots dos sistemes compten amb connectors per gestionar aquest mòdul.

2.4. Eines col·laboratives per generar documentació

Molts projectes de codi lliure utilitzen eines col·laboratives (principalment wikis) per redactar la seva documentació. Aquestes eines ofereixen molts avantatges i permeten començar a treballar molt ràpidament.

Les wikis poden funcionar mitjançant una base de dades o un sistema de fitxers. Per exemple, MediaWiki (www.mediawiki.org), programari utilitzat per la Viquipèdia, admet diferents tipus de bases de dades: MySQL, PostgreSQL, Oracle i SQLite. En canvi,

DokuWiki (www.dokuwiki.org) utilitza el sistema de fitxers i, per consegüent, no requereix accés a una base de dades per funcionar.

Entre els avantatges de treballar amb una wiki hi ha:

- La sintaxi de les wikis és molt simple i habitualment els editors de text incrustats inclouen eines per donar format al text (canviar l'estil de la font, afegir títols, inserir imatges, etc.).
- Com que es tracta de plataformes en línia no cal que els col·laboradors es descarreguin cap programari d'edició especial.
- Les wikis habitualment inclouen un sistema de control de versions, de manera que es poden revertir els canvis o consultar l'estat anterior d'un mateix document.
- Algunes wikis inclouen un sistema d'anotacions que permet als diferents col·laboradors parlar sobre els continguts sense haver de modificar-los per inserir els seus comentaris.
- Hi ha molts tipus de wiki gratuïtes.
- A l'hora de cercar alguna informació és molt més ràpid cercar a una wiki que en un manual.

Malauradament també hi ha inconvenients quan es treballa amb una wiki:

* És molt fàcil que el contingut acabe duplicat, per exemple, perquè s'ha creat més d'una pàgina per un mateix tema.
* Quan es produeixen canvis no s'acostuma a actualitzar la documentació.
* Exportar els continguts pot ser problemàtic, ja que convertir aquesta documentació a altres formats no és una tasca trivial. Hi ha connectors per fer conversions, però el nivell de personalització és força limitat.

Cal tenir en compte que en tractar-se d'eines que no són dissenyades específicament per treballar amb documentació, el programari no pot forçar els col·laboradors a utilitzar un format concret per a la redacció dels seus documents.

Les opcions més simples per regularitzar aquests documents són:

- Configurar l'editor per restringir els elements que es poden utilitzar (per exemple, nivells de les capçaleres, no permetre el canvi de color, fixar la mida de les fonts, etc.).
- Utilitzar plantilles amb l'estructura que han de tenir els diferents documents. D'aquesta manera, cada vegada que s'ha de generar un nou document es parteix de la plantilla concreta omplint només les dades específiques del document.
- Posar a disposició de tots els col·laboradors una llista d'indicacions o estil indicant com s'ha de redactar cada tipus de document. Idealment, si hi ha tipus de col·laboradors molt diferenciats, s'haurien de redactar guies especialitzades per a aquests usuaris.

S'ha de tenir en compte que no totes les wikis admeten l'ús de plantilles, així que si es vol utilitzar aquest sistema es recomana comparar les capacitats de les diferents wikis abans d'escollir-ne que no cobreixin totes les necessitats, ja que afegir noves funcionalitats en el futur o migrar els documents a un altre sistema pot resultar molt costós.

2.4.1. DokuWiki

Una de les wikis més populars és DokuWiki, ja que com que no requereix l'ús d'una base de dades externa pot instal·lar-se en pràcticament qualsevol servei d'allotjament web. Entre les seves característiques més destacables hi ha:

- Té una sintaxi simple.
- El nombre de revisions per pàgina és il·limitat.
- Usa el ressaltat de sintaxi mitjançant editors alternatius (per exemple, Ace Editor).
- El sistema d'espais de nom facilita la categorització de les pàgines i la navegació per la wiki.
- Es genera una taula de continguts automàticament per cada document.
- Usa un sistema de bloquejos per evitar que més d'un usuari modifiqui un document simultàniament.
- Usa un sistema de control d'accés (ACL) per gestionar l'accés a usuaris individuals o grups.
- És més ràpida que una wiki basada en bases de dades, ja que no ha de realitzar consultes.

DokuWiki està desenvolupada en PHP. Cal tenir en compte que tot i que aquest és un projecte viu, porta en funcionament des de l'any 2004 i una part del seu codi no utilitza PHP orientat a objectes, sinó com si fossin guions que s'executen d'una tacada. Un altre dels motius pels quals destaca DokuWiki és pel gran nombre de connectors i temes que es poden descarregar gratuïtament i que permeten adaptar fàcilment la wiki a cada necessitat. Podeu trobar extensions en: https://www.dokuwiki.org/extensions

2.4.2. Wikis a GitHub

Com a exemple de la importància de les wikis com a eina col·laborativa per generar documentació en projectes de programari lliure hi ha GitHub (www.github.com). Es tracta d'un lloc web que ofereix els seus serveis per allotjar projectes de programari utilitzant el sistema de control de versions Git.

Aquests serveis són gratuïts per als projectes de programari lliure i repositoris públics, fet que permet a qualsevol persona afegir els seus repositoris lliurement, o inclús gaudir de repositoris privats si paga una quota mensual.

Aquest lloc web també ofereix dos sistemes per ajudar a documentar aquests projectes:

- Fitxer README: a cada directori d'un projecte es pot incloure un fitxer anomenat README (normalment només s'afegeix al directori arrel), que acostuma a incloure informació general sobre el projecte, com instal·lar-lo i com configurar-lo.
- Servei de wikis: cada repositori té integrada una wiki, de manera que es pot accedir a tota la documentació del projecte sense sortir de GitHub.

Tant el fitxer README com la wiki poden ser editats directament des de GitHub i tots dos suporten el format del llenguatge de marques Markdown.

3. Sistemes de control de versions

El control de versions permet mantenir un registre de canvis en un conjunt de documents al llarg del temps. Aquest tipus de control no es limita només al desenvolupament de maquinari, sinó que s'ha aplicat durant molts anys a la gestió de documents en paper, on s'identifica cada versió del document amb un nombre, la data i hora en la qual s'ha generat la revisió i el nom de la persona que ha fet els canvis.

En enginyeria del programari el **control de versions o revisions** és la pràctica que proporciona control sobre els canvis al codi font d'un programa. Habitualment aquest control es du a terme utilitzant eines especialitzades com són CVS, Mercurial, Git, etc.

Aquestes eines tradicionalment s'executen des de la línia d'ordres, però la majoria de sistemes de control de versions moderns proporcionen també una interfície gràfica i poden integrar-se amb els entorns de desenvolupament integrats (Integrated Development Environment o IDE en anglès) més populars.

S'ha de tenir en compte que els sistemes de control de versions també es troben integrats en altre tipus de programari, com per exemple els documents col·laboratius (Google Drive) o les wikis (Viquipèdia i DokuWiki), que implementen els seus propis sistemes de control de versions per poder revisar els canvis i restaurar versions anteriors.

3.1. Introducció als sistemes de control de versions

Disposar d'un sistema de control de versions resulta imprescindible a l'hora d'afrontar la implementació de qualsevol aplicació mínimament complexa, ja que proporcionen als desenvolupadors l'opció de desfer canvis i tornar a versions anteriors del desenvolupament de manera fàcil.

Aquests sistemes estan pensats per a treballar principalment amb fitxers de codi, és a dir, fitxers de text pla.

3.1.1. Repositori

El lloc on es desen els conjunts de canvis s'anomena **repositori**. Normalment es tracta d'un directori on es troben tots els fitxers del projecte i les dades amb l'historial de canvis, cosa que permet fer una còpia d'aquests fitxers en qualsevol moment concret, determinat pel seu número de versió. Cada cop que s'afegeixen canvis al repositori es crea una nova entrada a l'historial de canvis on s'inclou el número de versió i a partir d'aquest número és possible restaurar aquest estat.

Cal distingir entre els repositoris locals i els repositoris remots. Segons l'eina utilitzada, un repositori remot es troba en un servidor independent o es tracta d'un repositori local d'un altre equip. En qualsevol dels dos casos el concepte més important és que els repositoris poden sincronitzar-se de manera que diferents usuaris poden treballar en el mateix projecte i els mateixos fitxers simultàniament.

3.1.2. Tronc i branques

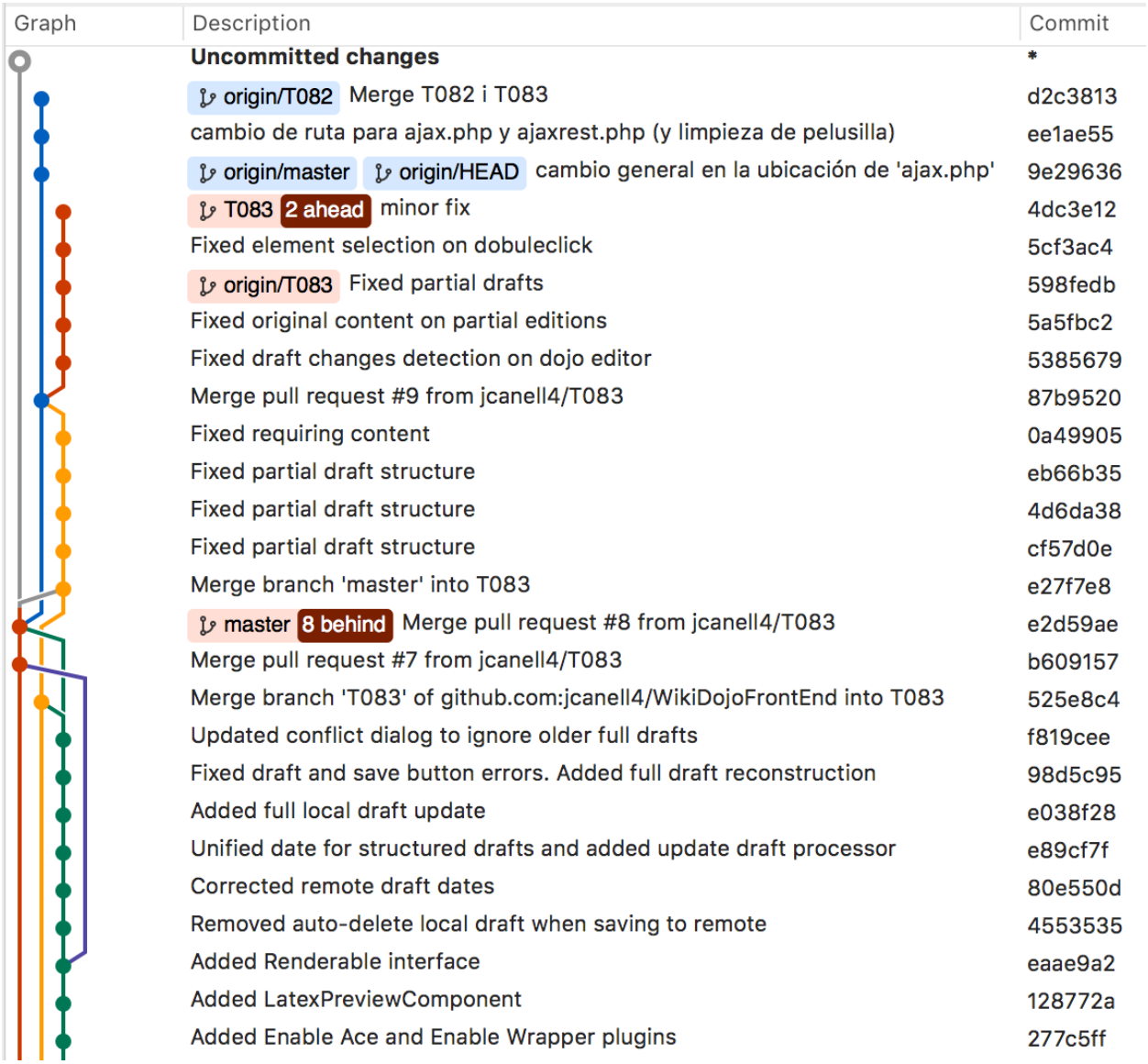
L'estructura d'un sistema de control de versions es pot interpretar com un arbre, on el tronc és on es pugen els canvis realitzats a l'aplicació i poden crear-se branques per

treballar en noves característiques o solucions d'errors que més endavant s'afegiran al tronc. D'aquesta manera es pot treballar de forma segura en una branca sense que els canvis del treball en progrés afecten a la branca principal.

Hi ha diversos casos en què una branca no torna a fusionar-se amb el tronc. Per exemple, es podria crear una branca per comprovar si una funcionalitat és viable o si una implementació diferent d'una funcionalitat existent millora el rendiment i, segons el resultat, fer la fusió amb la branca principal o descartar-la.

En altres casos pot crear-se una branca per mantenir els canvis d'una versió anterior (per exemple, la versió 1 del programari) mentre que al tronc es treballa amb una nova versió (per exemple, la versió 2). D'aquesta manera, tot i que a la branca principal es treballa amb una nova versió, si es detecten errors a la versió 1 poden solucionar-se i pujar aquests canvis a la branca corresponent a la versió 1 (per distribuir-los als clients que encara utilitzen aquesta versió).

No hi ha límit en el nombre de branques que poden crear-se. Per exemple, un programari podria tenir 4 branques per mantenir al mateix temps 4 versions diferents. A la figura podeu veure la representació d'un repositori on la línia de color gris és el tronc i les línies blava, roja, groga i verda representen diferents branques.



3.1.3. Tipus de sistemes de control de versions

Segons la seua arquitectura, els sistemes de control de versions poden dividir-se en els següents tipus:

- **Sistemes centralitzats**: fan servir una arquitectura client-servidor on tots els clients connecten amb el servidor per pujar o baixar els canvis, i l'equip local no guarda cap historial dels canvis.
- **Sistemes distribuïts** o **descentralitzats**: cada usuari té una còpia completa de tots els fitxers, l'historial de canvis i les pujades i baixades es realitzen entre els repositoris de cada usuari per sincronitzar-se entre ells.

Els sistemes centralitzats presenten l'avantatge de reduir el pes del projecte en els clients, ja que aquests només contenen una còpia dels fitxers i no dels canvis anteriors. Per contra, si falla el servidor cap dels clients podrà ni baixar els canvis ni pujar els nous. En el pitjor dels casos, si es perden les dades del servidor, es perdria tota la informació del repositori (si no hi ha còpia de seguretat).

En el cas dels sistemes distribuïts, si un client falla no hi ha cap problema, perquè pot recuperar-se el repositori complet juntament amb l'historial de canvis de qualsevol altre equip, de manera que només es perdria la informació no pujada des del client que ha fallat. L'inconvenient és que com que no hi ha un servidor central s'han de sincronitzar tots els repositoris entre ells o seleccionar un repositori com a central i fer que tots els repositoris facen servir el mateix per pujar i baixar els canvis.

GitHub (github.com) és el servei d'allotjament de repositoris Git més popular i ofereix integració amb múltiples eines de desenvolupament.

Actualment, l'opció més popular és una mescla entre els dos sistemes: utilitzar un servidor central com GitHub, ja que es tracta d'una plataforma molt segura, i fer servir als clients un sistema distribuït (Git).

Una altra característica important per a diferenciar entre tipus de controls de versions és el sistema de concurrència utilitzat:

- **Fusió** (*merge*): quan es produeix un conflicte perquè s'han fet canvis sobre un fitxer que ha estat modificat, es resol fusionant els fitxers (automàticament o manualment).
- **Bloqueig** (*lock*): quan un usuari modifica un fitxer, aquest es bloqueja de manera que cap altre usuari pot modificar-lo.

Cal tenir en compte que alguns dels sistemes de control de versions ofereixen totes les opcions, és a dir, poden utilitzar-se com a sistema centralitzat o distribuït i permetre l'ús dels models de concurrència de fusió i de bloqueig.

3.1.4. Terminologia

Cal tenir en compte que no tots els sistemes de control de versions utilitzen els mateixos termes per referir-se als mateixos conceptes. Per aquest motiu és important familiaritzar-se amb la terminologia. A continuació, trobareu una llista amb el nom dels termes més comuns i el nom o noms en anglès relacionats:

- **Repositori** (*repository* o *depot*): fa referència al lloc on es guarden els fitxers actuals i l'històric de canvis.
- **Tronc** (*trunk* o *master*): és la branca principal d'un repositori de la qual ixen la resta de branques.
- **Branca** (*branch*): és una bifurcació del tronc o branca mestra de l'aplicació que conté una versió independent de l'aplicació i a la qual poden aplicar-se canvis sense que afecten ni al tronc ni a altres branques.
- **Cap** (*head* o *tip*): fa referència a la versió més recent d'una determinada branca o del tronc. El tronc i cada branca tenen el seu propi cap, però per referir-se al cap del tronc de vegades s'utilitza el terme *HEAD*, en majúscules.
- **Còpia de treball** (*working copy*): fa referència a la còpia local dels fitxers que s'han copiat del repositori, que és sobre la qual es fan els canvis (és a dir, s'hi treballa) abans d'afegir aquests canvis al repositori.

- **Bifurcació** (*fork*): consisteix a crear un nou repositori a partir d'un altre. Aquest nou repositori, al contrari que en el cas de la clonació, no està lligat al repositori original i es tracta com un repositori diferent.
- **Clonar** (*clone*): consisteix en crear un nou repositori que és una còpia idèntica d'un altre, ja que conté les mateixes revisions.
- **Pujar** (*commit* o *check in*): és afegir els canvis locals al repositori. Cal destacar que no els envia al servidor; els canvis queden emmagatzemats al repositori local que s'ha de sincronitzar.
- **Baixar** (*check out*): és copiar a l'àrea de treball local una versió des d'un repositori local, un repositori remot o una branca diferent.
- ***Pull***: és l'acció que copia els canvis d'un repositori (habitualment remot) en el repositori local. Aquesta acció pot provocar conflictes.
- ***Push*** o ***fetch***: són accions utilitzades per afegir els canvis del repositori local a un altre repositori (habitualment remot). Aquesta acció pot provocar conflictes.
- **Canvi** (*change* o *diff*): representa una modificació concreta d'un document en el control de versions.
- **Sincronització** (*update* o *sync*): és l'acció de combinar els canvis fets al repositori amb la còpia de treball local.
- **Conflicte** (*conflict*): es produeix quan s'intenten afegir canvis a un fitxer que ha estat modificat prèviament per un altre usuari. Abans de poder combinar els canvis amb el repositori s'haurà de resoldre el conflicte.
- **Bloqueig** (*lock*): alguns sistemes de control de versions en lloc d'utilitzar el sistema de fusions el que fan és bloquejar els fitxers en ús, de manera que només pot haver un sol usuari modificant un fitxer en un moment donat.
- **Fusionar** (*merge* o *integration*): és l'acció que es produeix quan es volen combinar els canvis d'un repositori local amb un remot i es detecten canvis al mateix fitxer en tots dos repositoris i es produeix un conflicte. Per resoldre aquest conflicte s'han de fusionar els canvis abans de poder actualitzar els repositoris. Aquesta fusió pot consistir en descartar els canvis d'un dels dos repositoris o editar el codi per incloure els canvis del fitxer en les dues bandes. Cal destacar que és possible que un mateix fitxer presente canvis en molts punts diferents que s'hauran de resoldre per poder donar la fusió per finalitzada.
- **Versió** (*version* o *revision*): és el conjunt de canvis en un moment concret del temps. Es crea una versió cada vegada que s'afegeixen canvis a un repositori.
- **Etiqueta** (*tag*, *label* o *baseline*): permet afegir una etiqueta a una pujada per poder identificar aquesta pujada concreta d'una forma més entenedora. Per exemple, es pot etiquetar la primera versió d'un programari (1.0) o una versió en la qual s'ha solucionat un error important.
- **Tornar a la versió anterior** (*revert*): descarta tots els canvis produïts a la còpia de treball des de l'última pujada al repositori local.

3.1.5. Programari de control de versions

Hi ha molts sistemes de control de versions, tant gratuïts com de pagament. Entre les opcions més populars, per ordre de popularitat, hi ha:

- **Git**: és el programari més popular amb diferència. Es tracta d'un sistema distribuït, fa servir el model de concurrència de fusió i és gratuït.
- **Subversion** (SVN): es tracta d'un sistema centralitzat, permet fer servir el sistema de fusió o bloqueig i és gratuït.
- **Team Foundation Server** (TFS): és un programari desenvolupat per Microsoft que pot utilitzar les arquitectures centralitzades o distribuïdes i fer el model de fusió o bloqueig. És gratuït per a equips petits i projectes de codi lliure, mentre que en altres casos cal pagar una subscripció.
- **Mercurial**: es tracta d'un sistema distribuït, fa servir el model de fusió i és gratuït.

Com que tant Git com GitHub ofereixen una integració molt bona amb el programari de desenvolupament (entorns de desenvolupament i eines de gestió de projectes), aquesta combinació s'ha tornat molt popular i és utilitzada per projectes de programari lliure, empreses i institucions.

3.2. Utilització de Git

Git va ser creat per Linus Torvalds, l'any 2005, per al desenvolupament del nucli de Linux.

Git és d'un programari de control de versions que utilitza un sistema distribuït i, per consegüent, cada usuari que clona un repositori obté una còpia completa dels fitxers i l'historial de canvis.

Tot i que Git és un sistema distribuït, pot utilitzar-se una còpia del repositori en un servidor de manera que tots els usuaris pugen i baixen els canvis d'aquest repositori per facilitar la sincronització.

Cal tenir en compte que Git és un programari molt complex i inclou moltes característiques avançades per a la gestió de revisions i la visualització dels canvis. En aquests materials només es mostren les accions més habituals per poder treballar amb aquesta eina.

3.2.1. Instal·lació

Git acostuma a estar instal·lat a totes les distribucions de Linux.

Git es troba disponible a la majoria de plataformes, incloent Linux, Windows, macOS i Solaris. A la pàgina oficial podeu trobar l'enllaç per descarregar-lo per als diferents sistemes operatius.

Tot i que Git ofereix una interfície gràfica, en aquests materials s'utilitza mitjançant la línia d'ordres.

El procés d'instal·lació en qualsevol sistema operatiu és molt senzill. Un cop instal·lat, obriu la finestra del símbol del sistema (o la terminal, segons el sistema operatiu) i escriviu:

```
git --version
```

3.2.2. Operacions bàsiques

En resum, les ordres bàsiques de Git per treballar amb repositoris locals són les següents:

- **git init**: inicialitza un repositori.
- **git add** : afegeix elements de la còpia de treball al control de versions.
- **git rm –cache**: elimina elements del control de versions.
- **git status**: mostra l'estat de la còpia de treball.
- **git commit**: puja els canvis de la còpia de treball sota el control de versions al repositori local.
- **git log**: mostra la llista de versions pujades al repositori local.
- **git clone**: copia un repositori remot, no cal inicialitzar-lo.

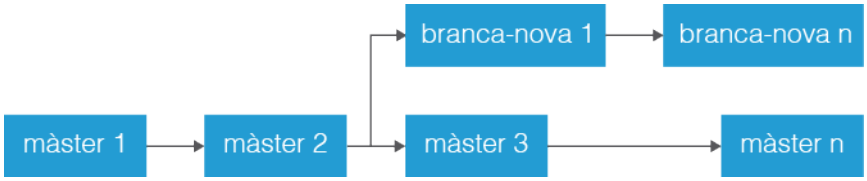
3.2.3. Operacions avançades

És molt habitual quan es treballa en control de versions haver de crear noves branques, de manera que al tronc es troba la versió actual del programari i a les branques es desenvolupen noves funcionalitats, es fa el manteniment de versions anteriors o se solucionen errors.

En alguns casos, com la implementació de noves funcionalitats i la solució d'errors, aquestes branques es fusionen amb el tronc un cop s'ha finalitzat amb èxit la tasca.

Per crear una nova branca es fa servir l'ordre git branch.

Representació de branques a Git:



En el cas que vulgueu integrar els canvis fets en una branca amb el tronc (per exemple, perquè s'ha finalitzat la implementació de la nova funcionalitat o s'ha corregit l'error pel qual es va crear la branca), heu de fer servir l'ordre git merge.

De vegades interessa tornar a una versió anterior del control de canvis. Per fer-ho es pot utilitzar l'ordre checkout, però en lloc d'indicar una branca s'indica l'identificador de la versió. Com ara:

```
git checkout 3f93651
```

Per tornar a la versió més actual només cal entrar l'ordre git checkout, especificant una branca (per exemple, master):

```
git checkout master
```

Git permet afegir etiquetes a les versions de manera que es pot distingir entre les versions habituals i les que representen algun fet especial, com per exemple una versió estable del producte o alguna fita concreta (la inclusió d'alguna funcionalitat important).

Per afegir una etiqueta a l'última versió afegida al repositori s'utilitza l'ordre git tag -a. Per exemple:

```
git tag -a v1.0 -m "Primera versió"
```

L'opció -a indica el text de l'etiqueta i l'opció -m permet afegir informació addicional.

Per llistar totes les etiquetes d'un repositori s'utilitza l'ordre git tag sense cap opció.

De vegades pot interessar descartar tots els canvis realitzats a la còpia de treball i tornar a la revisió actual. En aquest cas es pot fer servir l'ordre:

```
git reset --hard
```

Aquesta ordre descarta tots els canvis i tots els fitxers sota el control de canvis són restablerts. Aquesta acció és coneguda com a "revertir els canvis" en alguns entorns.

Quan es treballa amb repositoris remots (per exemple, quan es clona un repositori) es poden descarregar els canvis que s'han portat a terme en el repositori remot amb l'ordre git pull. Aquesta ordre descarrega els canvis i fa una fusió automàtica amb la còpia de treball. Aquesta acció pot provocar conflictes que s'han de resoldre abans de poder pujar els canvis al servidor.

En el cas de voler pujar els canvis del repositori local al repositori remot l'ordre que s'ha d'utilitzar és git push. Per executar aquesta ordre primer heu de fer un *pull* per fusionar els canvis al servidor remot amb la còpia de treball. En cas contrari, si s'han produït canvis al repositori remot, l'ordre *push* és rebutjada.

Per afegir un repositori remot heu de fer servir l'ordre git remote add, indicant el nom del repositori remot i l'URL corresponent. Per exemple:

```
git add remote add xat https://github.com/xaviergaro/client-servidor-xat
```

Aquesta ordre afegeix com a repositori remot l'URL https://github.com/xaviergaro/client-servidor-xat amb *xat* com a nom curt.

Per fer un *push* al servidor remot heu d'indicar el repositori d'origen (per al repositori local és origin) i el repositori de destí. Per exemple, per fer un *push* des del repositori local al repositori remot *xat* l'ordre seria la següent:

```
git push origin xat
```

Per fer un *pull* només cal indicar el nom curt del repositori remot:

```
git pull xat
```

En resum, les ordres avançades de git són:

- **git branch**: crea noves branques o llista les branques del repositori.
- **git checkout**: canvia la còpia de treball a la branca o versió indicada.
- **git diff**: mostra els canvis que s'han afegit en una versió.
- **git log**: mostra la llista de versions per la branca activa.

- **git merge**: fusiona els canvis entre dues branques.
- **git tag**: afegeix una etiqueta a una versió.
- **git show**: mostra informació sobre la versió indicada.
- **git reset –hard**: reverteix els canvis a la còpia de treball.
- **git pull**: puja els canvis del repositori local a un repositori remot.
- **git push**: baixa els canvis d'un repositori remot al repositori local.
- **git remote**: afegeix un repositori remot o llista els repositoris remots enllaçats amb el repositori local.
- **fitxer .gitignore**: permet afegir una llista de fitxers i directoris per excloure del sistema de control de versions.

3.2.4. Entorn gràfic

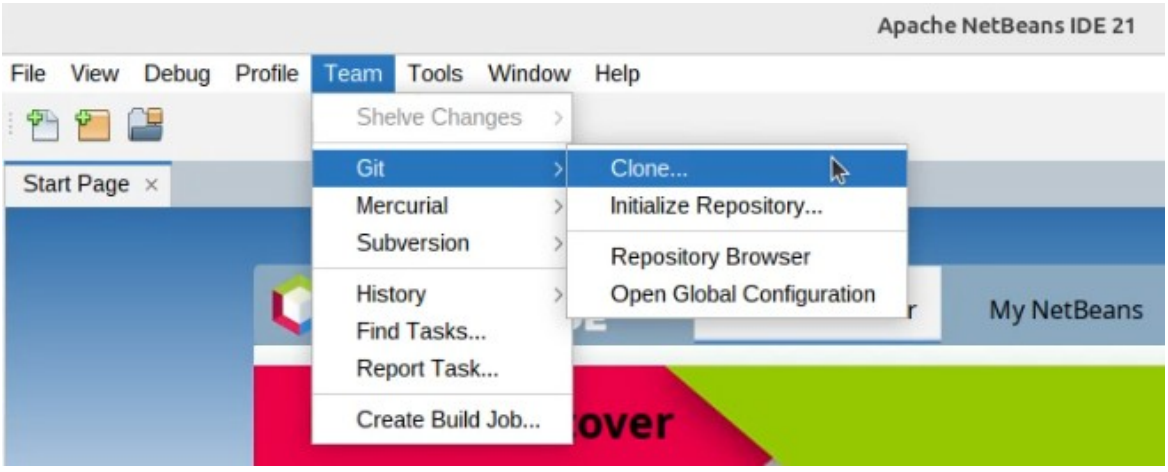
Per a Linux es poden trobar clients gràfics com SmartGit o GitKraken.

SourceTree és un client gràfic gratuït per a Git desenvolupat per Atlassian per a les plataformes Windows i macOS (no es troba disponible per a Linux). Aquest programari és el més popular dels clients de Git actualment al mercat.

3.2.5. Integració amb entorns de desenvolupament integrats: Netbeans

Tots els entorns de desenvolupament integrats (i alguns editors de text per a desenvolupadors) inclouen l'opció de gestionar el control de versions dintre del mateix programa. En aquesta secció es descriu com integra NetBeans el control de versions amb Git.

A NetBeans hi ha totes les opcions de Git a l'opció *Team> Git* del menú principal, com podeu veure en la figura:



Des d'aquest menú podeu inicialitzar un repositori per al projecte actual o clonar un repositori existent. Per clonar un repositori existent no és necessari crear un projecte, aquest es crea a partir del repositori clonat. Per clonar un repositori existent heu de seleccionar *Team> Git> Clone* al menú principal. Seguidament heu d'indicar la ruta al directori que voleu clonar (pot ser una URL, si es tracta d'un repositori remot).

A continuació heu de marcar les caselles corresponents a les branques que voleu clonar.

Finalment heu de decidir el directori de destí, el nom del repositori clonat i quina és la branca activa.

Una vegada clonat o inicialitzat un repositori es poden portar a terme totes les accions relacionades amb Git fent clic sobre un dels fitxers amb el botó dret del ratolí (o des de l'opció *Team* del menú principal.

Alguns entorns de desenvolupament inclouen icones amb les accions més comunes de Git (per exemple, Eclipse o la família de IDE derivades d'IntelliJ), però en general el funcionament d'aquestes eines és molt similar. Totes inclouen accés a Git des d'un menú contextual i des de la barra de menú principal.

3.3. Utilització de Github

GitHub (github.com) és un servei d'allotjament de repositoris Git que compta amb més de 10 milions d'usuaris. Ofereix tota la funcionalitat de Git, a més d'oferir serveis propis com són l'edició de fitxer en línia, la gestió d'errors, possibilitat de documentar els projectes mitjançant una wiki inclosa al repositori o la gestió d'usuaris.

Hi ha dos tipus de repositoris a GitHub:

- Públics: tothom pot visualitzar-los i descarregar-los, sense necessitat de crear un compte a GitHub. Aquests repositoris són gratuïts, i qualsevol usuari registrat pot crear-los.
- Privats: només els membres de l'equip i els usuaris amb permisos poden visualitzar, baixar i pujar canvis al repositori. Aquests repositoris estan limitats als comptes de pagament o d'estudiant (requereixen una adreça de correu universitari vàlida).

Les wikis incloses als repositoris de GitHub compten amb el seu propi control de versions i poden clonar-se mitjançant Git.

GitHub inclou característiques de xarxa social com ara notificacions, llistes de seguidors, opció de subscriure's als repositoris per fer un seguiment dels canvis o marcar repositoris com a favorits. Tot i que la plataforma no proporciona cap sistema de missatgeria entre usuaris, alguns usuaris afegeixen la seua adreça de correu electrònic al seu perfil i és possible comunicar-se amb els administradors d'un repositori mitjançant el sistema de gestió d'errors (*issues*) o la wiki que es pot incloure al repositori.

Per treballar amb GitHub es pot utilitzar la línia d'ordres de Git, es pot descarregar algun client gràfic com SourceTree o Github Desktop o es pot treballar directament des d'un IDE integrat amb Git.

Per començar a treballar amb els repositoris de GitHub com a repositoris remots cal crear un compte a la plataforma. En cas contrari, es poden clonar els repositoris públics però no es poden pujar els canvis.

Una vegada creat un compte, podeu crear nous repositoris des de la pàgina fent clic al botó *New repositori* .

El fitxer README admet el format Markdown i acostuma a incloure informació detallada sobre el repositori i instruccions d'instal·lació.

A la secció d'opcions generals hi ha els següents apartats:

- **Quines característiques es volen activar**: activar la wiki, restriccions d'editors, gestió d'errors i gestió de projectes.
- **Com es volen fusionar els canvis**: llista diferents mètodes de fusió de canvis.
- **Limitació temporal d'interacció**: permet imposar una limitació d'interacció temporal amb el repositori per diferents tipus d'usuaris.
- **GitHub Pages**: configura un servei que permet crear un lloc web per al repositori o el projecte.
- **Zona de perill**: permet canviar el tipus de repositori (públic o privat), transferir la propietat o eliminar el repositori.

Respecte a la gestió d'usuaris, GitHub permet afegir **col·laboradors** a un repositori fent clic a l'opció *Collaborators* de la barra esquerra de la secció de configuració. En aquesta secció es mostra la llista de col·laboradors actuals i un botó per enviar una invitació a altres usuaris mitjançant el seu nom d'usuari o adreça de correu electrònic. Aquests col·laboradors podran visualitzar el repositori encara que siga privat i podran pujar canvis al repositori remot. Cal destacar que l'administrador del repositori no té cap control sobre què pugen els col·laboradors, així que cal tenir-ho en compte a l'hora d'afegir col·laboradors a un repositori.

En el meu canal podeu trobar un mini tutorial sobre: ***Connectar un repositori Git local amb Github*** (https://youtu.be/Hl0GraQbk58)

O un tutorial molt recomanable sobre Git a: https://www.w3schools.com/git/