CMS introducció

Daniel Julià Lundgren

PID_00168277

UOC
Universitat Oberta de Catalunya

Índex

Introducció

Un dels aspectes que han influït més en l'evolució d'Internet en els últims anys ha estat l'aparició de diverses **eines de gestió de continguts** o **CMS** (de l'anglès *content management system*), que han permès a una gran part de la població mundial publicar continguts fàcilment i fer-los visibles en la Xarxa.

Un dels moments més importants d'aquesta evolució va ser l'expansió dels blogs personals que es va donar fa uns anys gràcies, en gran part, a l'aparició de multitud d'eines **de codi obert** i **gratuïtes** que permeten a qualsevol persona crear el seu propi blog sense tenir coneixements ni d'HTML ni de programació ni de disseny web.

El món de la publicació web, que fins llavors havia estat restringit a gent amb coneixements tècnics i que es donava normalment en els àmbits universitaris, es va estendre a gran part del gran públic.

En aquesta assignatura parlarem de les eines que han permès aquesta evolució, els CMS.

Un d'aquests CMS, el WordPress, que veurem aquí, està pensat específicament per a la creació i manteniment de blogs i és, probablement, el CMS més utilitzat actualment, no sols per a blogs, sinó també per a gairebé tot tipus de llocs web.

L'aparició de CMS de codi obert es correspon amb l'evolució natural d'Internet.

Els continguts que hi ha en el web han evolucionat des dels inicis, quan la majoria eren estàtics, de manera que a mesura que la quantitat d'informació va créixer, van necessitar ser dinàmics.

Un contingut **estàtic** és el que es crea "manualment" escrivint el contingut juntament amb el codi HTML amb un editor web com Dreamweaver. Una pàgina estàtica sempre es veu igual. En canvi, en un lloc web **dinàmic** les pàgines HTML es construeixen automàticament extraient el contingut d'una base de dades. Per a això, és necessari algun llenguatge en el servidor que permeti llegir de la base de dades (fer consultes SQL) i construir el codi de la pàgina HTML. Segons els paràmetres que es passin a la pàgina (en l'URL, utilitzant "?" i "&"), el contingut de la pàgina pot variar. Aquí apareix el concepte de **plantilla** o **tema** (*theme* en anglès). La mateixa pàgina ha d'estar pensada per a mostrar diferents continguts de diferent extensió.

Tradicionalment (especialment des de l'expansió de les pàgines web en Internet aproximadament en 1994 i fins a aproximadament l'any 2000), les pàgines web s'havien construït estàticament o dinàmicament, mitjançant programació en el servidor i la base de dades. Els llocs web amb continguts dinàmics tenien un panell d'administració, però normalment molt senzill i limitat només a opcions bàsiques.

El pas següent en aquesta evolució natural és cap als gestors de contingut de codi obert. Aquest pas es va fer al començament de la primera dècada del 2000. A mesura que les empreses utilitzaven més i més la tècnica de la creació dinàmica de pàgines web, es feia necessari poder reaprofitar el codi utilitzat en altres pàgines web, i diverses empreses van començar a oferir comercialment solucions de gestió de continguts propietàries. Després d'això, l'aparició d'eines de codi obert també potents només va ser qüestió de temps.

El fenomen del web 2.0 i el fet que cada vegada més els usuaris mateixos són generadors de contingut, s'han vist afavorits per aquesta filosofia basada en eines que permeten a qualsevol persona sense coneixements informàtics publicar continguts en el web.

Avui dia resulta difícil pensar en la gestió d'un web sense un bon gestor de continguts, ja que la immensa majoria de continguts d'Internet procedeix de CMS propietaris o de codi obert.

1. Definició de CMS

Un **gestor de continguts** o **CMS** (de l'anglès *content management system*) és una aplicació web que es fa servir per a crear, editar, gestionar i publicar contingut digital.

Normalment, els continguts es gestionen usant un navegador web convencional i amb un **panell d'administració**, al qual tenen accés l'administrador o els editors de la publicació.

Una característica comuna a tots els CMS és que **no són necessaris coneixements informàtics avançats** per a gestionar el contingut del lloc web. Un perfil d'editor de continguts inclou saber com es maneja l'eina, però en principi no inclou tenir coneixements d'HTML, CSS o llenguatges de servidor. Aquest és l'avantatge fonamental de qualsevol CMS.

En resum i citant la Wikipedia:

> "Un sistema de gestión de contenidos (en inglés *Content Management System*, abreviado CMS) es un programa que permite crear una estructura de soporte (o *framework*) para la creación y administración de contenidos, principalmente en páginas web, por parte de los participantes.
>
> Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Un ejemplo clásico es el de editores que cargan el contenido al sistema y otro de nivel superior (directorio) que permite que estos contenidos sean visibles a todo el público (los aprueba)."

Pàgina web

En la Wikipedia podem trobar una definició completa de CMS:
http://ca.wikipedia.org/wiki/sistema_de_gesti%C3%B3_de_continguts.

L'ús d'un CMS té multitud d'avantatges que detallarem a continuació. El més important d'aquests avantatges és, com dèiem, que no cal tenir coneixements elevats de programació per a instal·lar i configurar una aplicació i tenir un lloc web dinàmic que funcioni des de zero, de manera que amb prou feines hi ha "barrera d'entrada".

L'eclosió d'un gran nombre de CMS ha coincidit, no casualment, amb l'augment en general de les plataformes de **codi obert**. A més dels avantatges propis de qualsevol CMS, també n'hi ha d'altres d'intrínsecs a les opcions de codi obert en comparació amb les opcions propietàries.

1.1. Avantatges i desavantatges de les opcions de codi obert respecte a les propietàries

Els **avantatges** de les opcions de codi obert són:

- Cost nul o baix.

- Cost d'allotjament més baix que en les solucions propietàries, ja que les aplicacions del servidor també es basen en plataformes de codi obert, com LAMP (Linux, Apache, MySQL, PHP).

- Accés al codi font per a fer qualsevol modificació que considerem convenient.

- Possiblement la més important: en molts casos hi ha una **comunitat de desenvolupadors** que s'encarreguen de mantenir i d'actualitzar el codi permanentment.

- Generalment, la comunitat també crea i manté **extensions** (***plug-ins***) d'aquests CMS que ens permeten augmentar les funcionalitats.

- Solució de dubtes per mitjà d'aquesta comunitat.

Naturalment també hi ha alguns **desavantatges**:

- Si el nostre projecte no s'adapta a l'estructura de continguts o al cicle de treball (*workflow*) del CMS, els canvis poden ser molt difícils de fer.

- Depenem de la comunitat per a actualitzar la programació.

- Sovint la configuració del sistema és molt laboriosa i requereix bastant temps.

La quantitat de CMS disponibles és enorme: hi ha centenars, possiblement milers, de CMS diferents, la majoria descarregables directament des d'Internet. Per a facilitar una mica la cerca, han aparegut alguns directoris en què es pot consultar o buscar informació sobre CMS. Un d'aquests és el CMS Matrix (http://www.cmsmatrix.org/), servei complet de comparativa de característiques de gestors de continguts de codi obert.

1.2. Avantatges i desavantatges empresarials de l'ús d'un CMS de codi obert respecte a altres alternatives

Per a l'empresa, l'ús d'un CMS té diversos **avantatges**:

- Un CMS permet a l'organització o empresa concentrar-se en la creació de contingut o en la funció específica que es busca per al lloc web, en lloc del disseny o el desenvolupament de l'eina, que normalment ocuparia gran part del temps.

- Perfils no necessàriament tècnics de l'organització poden editar i publicar continguts. Normalment, perfils amb coneixements d'usuari poden ser formats sense gaires dificultats en la majoria dels CMS.

- Tots els continguts queden allotjats en un sol lloc centralitzat, en la base de dades, controlada per la mateixa empresa. Això, a més, té un potencial en el moment d'ampliar o d'interconnectar aquesta informació amb altres sistemes de l'organització (intranet, etc.).

- Es poden fer cerques sobre el contingut, filtrant per data, tipus de contingut, autor o qualsevol altre camp.

- Normalment, els continguts es poden programar per a ser publicats en una hora i una data determinades.

- Diversos autors o traductors poden treballar simultàniament sense dificultats. Per a cada perfil d'usuari, es poden definir uns permisos específics, per exemple, publicant el contingut només en el moment en què un perfil determinat (per exemple, el perfil "editor") ho aprova.

- Es poden ampliar funcionalitats com calendaris, fòrums, galeries de fotos, etc. Es tracta de funcionalitats que moltes vegades estan incloses en els CMS, directament o per mitjà d'ampliacions (connectors o *plug-ins*, extensions o mòduls segons la denominació específica de cadascun dels CMS).

- Un CMS es pot integrar amb altres sistemes de l'organització, com per exemple *newsletters* de clients o altres.

- Atès que el contingut i la presentació estan separats, canviar l'aparença (o *look and feel*) del lloc web és molt més fàcil que usant un sistema fet a mida. Aquesta és una de les característiques més importants que ha de tenir qualsevol CMS i, de fet, depèn en gran part de la recomanable separació entre HTML i CSS.

També hi ha alguns **desavantatges** que hem de valorar:

- Encara que sovint codi obert s'equipara a gratuït, pensar això és un error. Probablement s'haurà de fer una inversió inicial apreciable, tant en diners com en temps. Encara que podem abordar un projecte amb molt poc desenvolupament, sovint qualsevol CMS requereix molt temps per a la configuració, adaptació i integració de continguts.

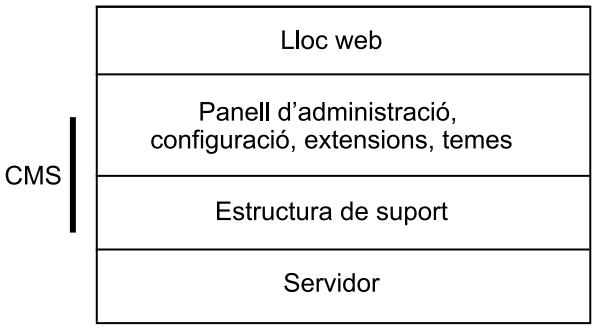
- Hi ha una corba d'aprenentatge inicial en l'equip que s'encarrega d'actualitzar els continguts.

- Usualment, una persona (o més) de l'organització s'ha de convertir en l'administrador de webs o *webmaster* del CMS per a prendre la responsabilitat del seu manteniment.

- Un CMS no converteix mals continguts en bons, ni converteix un mal escriptor/editor en un de bo. Els continguts han d'estar escrits correctament. El CMS només s'encarrega de gestionar els continguts, però no els pot corregir.

- Traslladar un CMS d'una plataforma a una altra pot ser complex i costós en temps i diners. Sempre cal intentar anticipar-se a noves necessitats per a poder fer les transicions d'una manera gradual.

El fet que un CMS tingui un gran nombre de funcionalitats automàticament també és la causa que de vegades sigui difícil complir exactament amb les funcionalitats del projecte que volem implementar. L'ajust de funcionalitats i de la presentació s'aconsegueix mitjançant la personalització amb la configuració en el panell d'administració a primer nivell, o mitjançant programació a segon nivell. Si això no és possible o és difícil, haurem de pensar en altres possibilitats, com l'ús d'un ***framework*** de desenvolupament d'aplicacions web. Un *framework* és un pas intermedi entre un sistema desenvolupat totalment des de zero i un CMS complet. En certa manera, un CMS és una evolució d'un *framework* afegint una capa d'administració i configuració.



CMS vist com a *framework* + panell d'administració

2. *Frameworks* enfront de CMS

Un ***framework*** és un conjunt d'eines i de biblioteques de programació sota una arquitectura determinada (normalment usen el paradigma model/vista/controlador, MVC) que faciliten el desenvolupament d'aplicacions web complexes.

Vegem un extracte de la definició de *framework* en la Wikipedia:

> "En el desarrollo de *software*, un *framework* es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de *software* concretos, con base en la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto."

Pàgina web

En la Wikipedia podem trobar una definició completa de *framework*:
http://es.wikipedia.org/wiki/Framework.

En certa manera, un *framework* es pot considerar la primera capa de qualsevol CMS. De fet, alguns CMS es poden usar com a *frameworks* de desenvolupament. Des d'un altre punt de vista, podem pensar que un *framework* està destinat al desenvolupador i, en canvi, un CMS a l'editor de continguts.

Així com és perfectament possible crear un lloc web amb un CMS sense haver d'escriure una sola línia de codi, amb un *framework* passa el contrari: **qualsevol funcionalitat implica un desenvolupament de programació**, ja que gairebé no hi ha eines visuals.

Pàgina web

Per a saber-ne més, trobareu una llista de *frameworks* en la Wikipedia: http://en.wikipedia.org/wiki/Content_management_framework.

2.1. El paradigma model/vista/controlador

En un projecte informàtic complex és molt important separar en mòduls les diferents parts del problema. Una separació molt freqüent és el denominat **paradigma model/vista/controlador** o **MVC**.

En la majoria de *frameworks* s'usa aquest sistema per a separar tres vessants de complexitat d'una aplicació:

- El **model** correspon a les **dades**. La connexió amb la base de dades, l'emmagatzematge, la recuperació i, també, la lògica de negoci, etc.

- La vista s'encarrega de la **presentació**, és a dir la interfície d'usuari (correspondria a la plantilla o al tema d'un CMS). Es basa en HTML i CSS però amb instruccions del llenguatge de servidor per a inserir els continguts dinàmics.

- El **controlador** es correspon amb la **interacció amb els usuaris** a força d'esdeveniments. Decideix a partir de paràmetres enviats a la pàgina web quin model i vista ha de cridar.

A efectes pràctics, en un *framework* basat en l'MVC els arxius del programa corresponents a cada nivell estan físicament separats en carpetes diferents. D'aquesta manera, s'aconsegueixen avantatges notables:

- Millora la claredat del codi.
- Es pot reutilitzar i ampliar més fàcilment.
- Es pot canviar la presentació de manera independent sense afectar la resta (només és necessari modificar la vista).

Pàgina web

En la Wikipedia podem trobar una definició més completa del MVC: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador.

Exemples típics de *frameworks* basats en l'MVC són:

- Ruby on Rails (basat en el llenguatge Ruby).
- Code Igniter (basat en PHP).
- Yii Framework (PHP).
- Zend Framework (PHP).
- CakePHP (PHP).
- Django (basat en el llenguatge Python).

Pàgina web

A http://www.phpframeworks.com/ es pot trobar una comparativa de *frameworks* basats en PHP.

L'**avantatge d'usar un *framework*** respecte a desenvolupar una aplicació des de zero és que **moltes funcionalitats** que probablement necessitem **ja estan implementades**.

Un exemple típic seria la gestió d'usuaris (autenticació, registre, etc.): un *framework* que tingui la gestió d'usuaris implementada ens permetrà estalviar-nos aquest feina.

Ara bé, en cas que vulguem implementar alguna d'aquestes funcionalitats en un *framework*, sempre ho haurem de fer mitjançant el **llenguatge de programació** de la plataforma, mitjançant una biblioteca, és a dir, programant a mà aquesta funcionalitat o editant arxius de configuració (a diferència de com es faria en un CMS).

2.2. Diferències entre un *framework* i un CMS complet

Un ***framework*** està enfocat a la programació. El desenvolupador usa una sèrie de **biblioteques** de codi que li permeten crear l'aplicació i que requereixen programació addicional. En canvi, en un **CMS** l'opció de desenvolupar codi és opcional i, en principi, no necessària ja que tot es configura visualment mitjançant el panell d'administració. Per augmentar les funcionalitats d'un CMS, normalment afegim **extensions** o **mòduls** que s'activen visualment (sense editar codi).

Un **CMS** té funcionalitats d'alt nivell (de la capa de presentació) que normalment no té un *framework*. Algunes d'aquestes funcionalitats poden ser:

- Panell d'administració per a totes les opcions del CMS.
- Activació de temes.
- Gestió dels menús de l'aplicació.
- Gestió dels blocs de continguts en les pantalles.

En un CMS totes aquestes característiques es poden activar sense necessitat de programar.

> Imaginem un cas concret: si hem creat una aplicació a partir d'un *framework* i més endavant volem afegir una funcionalitat, com la possibilitat d'autenticar-se (o *loginar-se*) mitjançant un compte de *Facebook*, només la podem incorporar a força de programar-la (bé a partir d'una biblioteca ja existent per al *framework*, si tenim sort, o bé des de zero, en cas contrari). Això no obstant, en un CMS, per a afegir aquesta funcionalitat probablement n'hi haurà prou d'activar un mòdul o connector (*plug-in*) i configurar-lo adequadament o, en cas que no n'hi hagi, desenvolupar-lo nosaltres (i idealment deixar-lo a la comunitat perquè posteriorment el reutilitzi i el mantingui).

En certa manera, podem considerar un *framework* com una capa inferior a un CMS complet. De fet, alguns CMS integren, implícitament o explícitament, un *framework* de base sobre el qual funciona la resta de funcionalitats.

D'alguna manera podríem fer la definició següent:

> *Framework* + funcionalitats d'alt nivell = CMS

Un ***framework*** és una eina molt útil per a un desenvolupador que hagi de crear una aplicació que, per les característiques que té, no s'adapta perfectament a cap CMS concret, o en cas que es prefereixi la flexibilitat total que permet implementar una aplicació usant biblioteques de programació en lloc del panell d'administració visual que s'usa en un CMS.

Entre les característiques genèriques que pot tenir un *framework* podem destacar les següents:

- **Model/vista/controlador**. Estructuració en la programació de l'aplicació en tres capes que corresponen a les dades, la presentació i la interacció amb l'usuari.

- **Seguretat**. Disposar de mecanismes de protecció contra atacs de *hackers*, *spam*, etc.

- **Mapatge d'URL netes** (*URL mapping*). Per exemple, mostrar en l'URL "llista_productes", en lloc d'"index.php?id=234124&op=234". Això és important per a l'accessibilitat i el posicionament en cercadors.

- **Accés a la base de dades**. Eines per a facilitar l'accés a la base de dades i fer-ho independentment del servidor concret de bases de dades (per exemple, MySQL, PostgreSQL, Oracle, etc.).

- ***Templates***. **Plantilles** (o temes) per a facilitar la separació de contingut i presentació.

- **Encauament** (*caching*) **de les pàgines per a augmentar-ne el rendiment**. Aquesta opció permet desar una versió creada pel llenguatge de servidor i la base de dades de la pàgina en la base de dades, de manera que en recuperar-la l'accés és molt més ràpid.

- **AJAX**. Petició i visualització de dades del servidor sense haver de recarregar la pàgina mitjançant JavaScript.

- **Autenticació d'usuaris**. Implementació d'inici de sessió (*login*), registre, dades de perfil, rols i permisos d'usuaris.

Atès que, com hem dit, un CMS es pot definir com un *framework* més una sèrie de capes addicionals, aquestes característiques també són les principals que trobarem en qualsevol CMS.

En resum, utilitzarem un ***framework*** quan necessitem flexibilitat total o quan el nostre projecte no s'ajusti bé a les funcionalitats predefinides en un CMS. En cas contrari, probablement la millor opció serà usar un **CMS**.

2.3. Avantatges d'utilitzar un CMS respecte a altres alternatives

Un CMS ens permet centrar-nos en la solució concreta de les necessitats del nostre projecte i oblidar-nos almenys parcialment del desenvolupament bàsic de l'aplicació. Entre els **avantatges** més importants tenim els següents:

- Possibilitat d'instal·lar i configurar l'aplicació sense tenir coneixements de programació en el servidor, HTML o CSS (encara que moltes vegades aquests coneixements són els que permeten ajustar perfectament la nostra solució als requisits i personalitzar les funcionalitats detalladament).

- Temps de desenvolupament més curt que implementant la solució des de zero o mitjançant un *framework* (sempre que s'ajusti a les funcionalitats del projecte).

- Llibertat total per a definir el disseny visual, ja que el CMS ha de separar la presentació del contingut. Normalment, la presentació es defineix a partir de temes (*themes*). Una personalització professional requereix crear o modificar plantilles mitjançant HTML, CSS i, probablement, PHP (o un altre llenguatge de servidor).

- Seguretat garantida en tenir una comunitat pendent d'oferir actualitzacions no sols per al nostre projecte, sinó també per a tots els que hi hagi basats en la mateixa plataforma. Si ens n'haguéssim d'encarregar nosaltres, aquest seria un procés laboriós, costós i continu en el temps. Probablement és un dels avantatges més importants.

- Possibilitat d'afegir extensions (segons el CMS es denominen *plug-ins* o *mòduls*) que augmenten les funcionalitats.

- Estructura de navegació coherent implementada per defecte. En mantenir la mateixa plantilla per a tots els continguts, s'afavoreix la coherència en la navegació.

- Validació HTML+CSS (almenys, l'hauria de garantir el mateix CMS).

- Crear diferents rols per al flux de treball en la publicació (per exemple, administrador, editor, membre de la comunitat, etc.).

- Disminuir el temps de desenvolupament. Moltes funcionalitats ja estan implementades.

- Disminuir el cost de desenvolupament.

- Facilitar la generació i edició de continguts.

- Manteniment. L'equip de desenvolupament i la comunitat d'usuaris del projecte de codi obert s'encarreguen de mantenir el codi de l'aplicació i d'oferir actualitzacions.

D'aquesta manera, ens pot interessar usar un **CMS** en cas que les especificacions d'aquest s'adaptin bé als requisits (funcionalitats) del nostre projecte.

> Per exemple, en cas que vulguem crear un blog, probablement una bona elecció serà el WordPress, ja que és un CMS dissenyat específicament per a aquesta necessitat. O, en cas que el nostre objectiu sigui crear un fòrum, podríem usar phpBB, un fòrum basat en PHP.

En el moment en què les funcionalitats siguin més diverses i no tan específiques, potser ens interessarà un CMS més flexible com Joomla o Drupal.

Però si les necessitats que tenim no s'ajusten realment a cap CMS del mercat, probablement utilitzar un ***framework*** i desenvolupar una aplicació a mida serà la millor solució, encara que en aquest últim cas serà necessari desenvolupar parts del codi de l'aplicació.

Observació

És important tenir en compte que tots aquests CMS són projectes "vius", en evolució contínua, i que per tant cal estar pendent d'actualitzacions i també de l'aparició de noves alternatives

Un dels **inconvenients** principals dels **CMS** és que, en estar pensats per a implementar un gran nombre de funcionalitats per defecte, si les nostres funcionalitats no s'ajusten perfectament a les que proporciona el sistema i la configuració tampoc no ens permet ajustar-les, l'esforç que haurem de fer (mitjan-

çant desenvolupament a mida) pot ser superior al que hauríem de fer partint d'una eina més senzilla com un *framework* o fins i tot partint des de zero. És per això que el primer pas ha de ser sempre avaluar a partir dels requisits de l'aplicació si aquests es poden satisfer amb un CMS determinat.

3. Ús dels CMS

Con qualsevol CMS, els passos que hauríem de seguir per a veure si és un candidat a solucionar el nostre problema són els següents:

- S'adapta a les necessitats i funcionalitats que volem implementar?
- Tenim disponibilitat en el nostre servidor de la plataforma tecnològica en la qual està basat?
- Podrem aconseguir per mitjà del panell d'administració l'estructura de continguts i el flux de navegació que es requereix?

Per a poder contestar a aquestes preguntes és necessari, per descomptat, tenir bons coneixements i experiència sobre les possibilitats dels CMS en qüestió.

En un bon CMS, el contingut i la presentació han d'estar totalment separats de manera que, en principi, podem dissenyar qualsevol presentació usant HTML/CSS i implementar-la dins del sistema. L'inconvenient sol ser més en l'estructura de continguts, en el flux de treball, en funcionalitats específiques, etc. que en la presentació.

3.1. Tipus de CMS i característiques

Cada CMS té uns requisits tècnics (plataforma tecnològica), està especialitzat en una funció específica i té una llicència d'ús determinada. Segons aquests punts de vista els podem classificar:

- Segons la **plataforma** (inclòs el llenguatge de programació emprat).
- Segons el tipus de **llicència** (propietat del codi: propietari o de codi obert).
- Segons el tipus d'**aplicació que volem desenvolupar** (fòrum, blog, portal, etc.).

3.1.1. Segons la plataforma

En un servidor web, normalment tenim tres components:

- El **servidor de pàgines web** (per exemple, Apache).
- El **llenguatge de programació** del servidor (per exemple, PHP).
- El **servidor de bases de dades** (per exemple, MySQL).

El component més restrictiu per a un CMS és el **llenguatge de programació** del servidor, ja que el codi del CMS està escrit en aquest llenguatge i, per tant, és una condició necessària per al seu funcionament. El servidor web i el de

la base de dades són, en molts casos, intercanviables (hem de consultar els requisits mínims de la plataforma per a assegurar-nos d'això); no obstant això, el llenguatge de servidor no ho és mai.

La majoria de servidors de bases de dades relacionals funcionen en SQL, un llenguatge de base de dades bastant estàndard, de manera que sovint és possible canviar de servidor de bases de dades sense grans canvis en el codi del CMS. De la mateixa manera, la tasca principal del servidor web, servir pàgines HTML, és independent de la resta de components, de manera que moltes vegades podem usar diferents servidors web (com l'Apache o l'MS IIS) sense problemes.

Alguns llenguatges de servidor sobre els quals hi ha múltiples CMS:

- ASP.NET (propietari Microsoft)
- Java
- PHP
- Ruby On Rails
- Python

Observació

En el mateix servidor podem tenir disponibles simultàniament diversos llenguatges, per exemple PHP i Java.

Observació

Normalment, els CMS de codi obert funcionen en plataformes de codi obert (com PHP), però no ha de ser així en tots els casos, ja que podem tenir perfectament un CMS de codi obert sobre ASP (sistema propietari) o un CMS propietari sobre PHP.

3.1.2. Segons la propietat del codi

Segons aquesta aproximació podem dividir les aplicacions de CMS en dos grans grups:

- **De codi obert**. Són les que permeten l'accés al codi font de manera que, si és necessari, es pot modificar qualsevol fragment o fins i tot crear noves aplicacions (nous CMS) a partir de la versió original.

- **De codi privatiu**. Només el seu creador o empresa poden desenvolupar o modificar el codi base de l'aplicació.

La majoria dels CMS que veurem aquí són de codi obert. Tots es regeixen per una llicència que especifica els termes i condicions sobre els quals es poden usar i modificar. Hi ha multitud de llicències, encara que algunes estan molt més esteses que altres, com per exemple la llicència GNU GPL (*general public license*), la més estesa entre els CMS que analitzarem.

Pàgina web

Per saber més sobre llicències de codi obert vegeu la llista de llicències per nom de l'Open Source Iniciative:
http://www.opensource.org/licenses/alphabetical.

3.1.3. Segons el tipus d'aplicació a desenvolupar

En el moment d'escollir un CMS, la primera decisió està determinada pel tipus d'aplicació que volem desenvolupar. Partir d'un CMS que s'ajusti a la funcionalitat principal del nostre projecte és crucial per a l'èxit de la seva implementació. Així, per exemple, hi ha CMS pensats per a diferents necessitats com:

- Creació i administració de blogs
- Portals d'empreses
- Entorns educatius
- Xarxes socials

Més endavant veurem detalladament alguns d'aquests tipus de projectes més comuns i els seus requisits, conjuntament amb una llista de CMS disponibles actualment adaptats a cadascun.

4. Components del servidor necessaris per a instal·lar CMS

Els tres components bàsics en qualsevol servidor d'aplicacions web (**servidor de pàgines web**, **llenguatge del servidor** i **servidor de base de dades**) funcionen sota un determinat **sistema operatiu**. Totes aquestes opcions es poden combinar entre elles i, a més, hi ha diverses versions, per la qual cosa és molt important assegurar-nos sempre dels requisits mínims del CMS que usarem.

Algunes de les diferents opcions que tenim per a cadascun dels components són:

- **Sistema operatiu**: Linux, MacOS, Windows, etc.
- **Servidor de pàgines web**: Apache (el més utilitzat), IIS (Microsoft), lighttpd, etc.
- **Llenguatge de servidor** (o servidor d'aplicacions): PHP, ASP, Java, Ruby, Python, etc.
- **Servidor de base de dades**: MySQL, PostgresSQL, Oracle, etc.

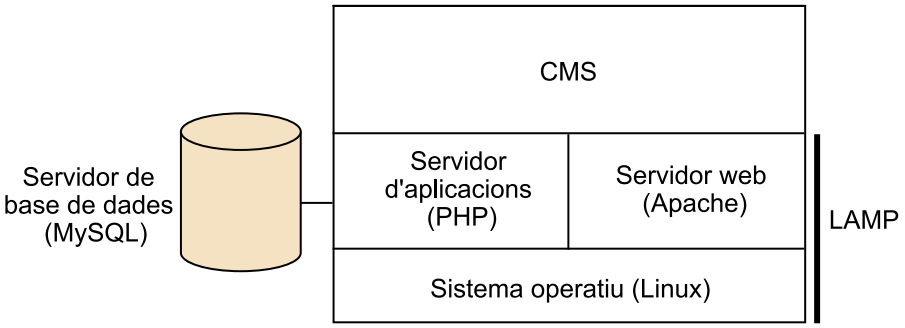
En realitat, el nostre CMS podria estar funcionant pràcticament en qualsevol combinació dels quatre components anteriors.

4.1. LAMP

La majoria dels CMS que veurem estan basats en **la plataforma LAMP**, que correspon als components del servidor web: **Linux+ Apache+MySQL+PHP**.

Tots aquests components són de **codi obert**, la qual cosa n'ha facilitat la difusió i l'ús en un gran nombre d'aplicacions.

- **Linux**: sistema operatiu de codi obert amb el qual funcionen actualment una bona part dels servidors web a Internet.
- **Apache**: servidor web de codi obert, també dels més usats actualment, encara que n'hi ha molts d'altres.
- **MySQL**: servidor de base de dades SQL de codi obert, actualment propietat de l'empresa Oracle.
- **PHP**: extensió del servidor web per a poder interpretar el llenguatge.



Components del servidor en una configuració LAMP

En general, és possible instal·lar aquests CMS en altres plataformes, com Windows o Mac OS, i en altres servidors web (diferents d'Apache). El que sí sol ser un requisit important en els CMS que veurem és **PHP+MySQL**.

En el cas d'un lloc web construït a força de pàgines estàtiques només necessitaríem el servidor de pàgines web (no PHP ni MySQL ni altres llenguatges de servidor i bases de dades).

4.2. Servidors web

Ens podem referir al servidor web com a *servidor HTTP* (pel protocol) o com a *HTTP daemon* (abreujat *httpd*), és a dir, *dimoni d'HTTP*, perquè funciona en mode *background* sense presència visual.

El servidor web més utilitzat actualment és **Apache**, una aplicació de codi obert que forma part d'un grup de projectes més ampli de la Fundació Apache i que, com hem dit, és el component de servidor web de l'anomenada *plataforma LAMP*.

Usant només el servidor web podem tenir pàgines en format HTML, text net, imatges o, en general, qualsevol arxiu que no necessiti ser interpretat. Per tant, és l'únic component que necessitem en el servidor per a tenir un lloc web amb **pàgines estàtiques**. No obstant això, aquests servidors es poden ampliar activant determinats mòduls, com són el que permet **pàgines dinàmiques** en PHP (mod_php) o connexions segures mitjançant SSL (mod_ssl), entre molts altres.

Pàgines web

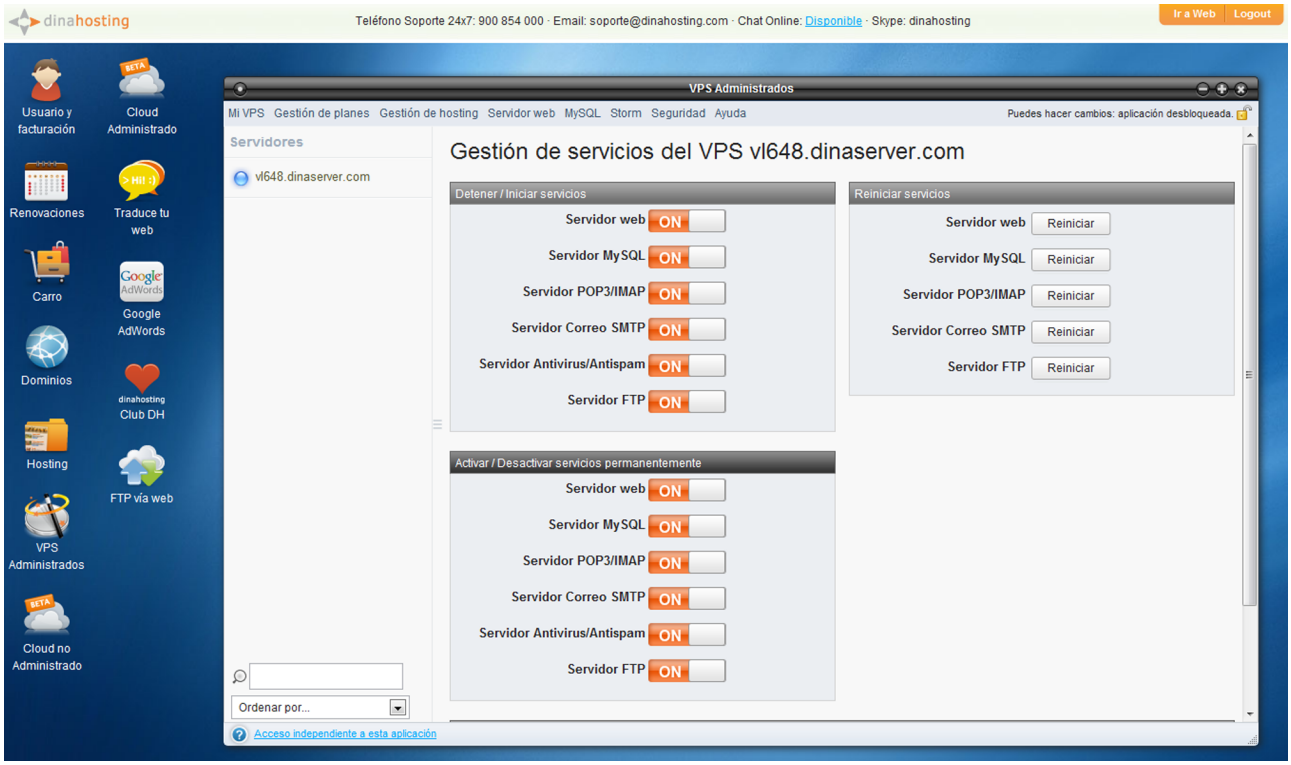
Sobre Apache:
http://httpd.apache.org.
Altres servidors web en la Wikipedia:
http://es.wikipedia.org/wiki/Servidor_HTTP_Apache.
http://en.wikipedia.org/wiki/Comparison_of_web_server_software.

4.3. Panell de l'allotjament web

Per accedir al nostre servidor normalment farem servir un **panell d'administració** del mateix servei d'allotjament. En el panell de l'allotjament tindrem accés a informació important com:

- Tipus de plataforma
- Bases de dades (i comptes associats)
- Configuració del servidor

- Gestió de comptes de correu i FTP



Exemple de tauler de control d'un servei d'allotjament web (*hosting*).

Les opcions i l'aparença del tauler de control varien segons l'empresa d'allotjament, i poden ser molt senzilles (o no haver-n'hi) o molt completes, com les de la imatge anterior.

En general, les funcions que necessitarem per a poder instal·lar un CMS són:

- Accés per FTP al servidor (compte associat).

- Accés a la base de dades (per exemple, mitjançant "phpMyAdmin" si usem PHP+MySQL). Necessitarem el nom i el compte associat a la base de dades.

- L'adreça pública des de la qual puguem accedir al contingut del web (per a poder-lo provar). Per exemple, "elmeullocweb.com" pot correspondre a l'adreça del servidor FTP o és possible que aquest es trobi en un altre URL. Ens hem d'assegurar d'aquest punt.

4.4. Accés per FTP

Per a accedir al servidor remot normalment usarem un programa de FTP. FTP significa 'file transfer protocol' o 'protocol de transferència d'arxius'; com el seu nom indica, no és més que un sistema per a copiar arxius d'una màquina a una altra, normalment entre el nostre PC de treball i el servidor web.

Recomanació

Recomanem usar FileZilla (http://filezilla-project.org/), un programa d'FTP de codi obert i disponible en totes les plataformes.

Mitjançant el protocol FTP tenim accés a tots els arxius (i directoris) del servidor remot, i fins i tot es poden canviar els permisos dels arxius i els directoris.

Observació

En molts CMS alguns directoris necessiten permisos d'escriptura perquè l'aplicació pugui funcionar, i això és una causa important de possibles problemes de funcionament.

En alguns casos és necessari tenir accés total al servidor, no solament als seus arxius, sinó també a l'execució de programes; amb aquest propòsit podem usar algun programa que permeti l'accés mitjançant Telnet/SSH. Gràcies al **protocol Telnet** disposem d'un terminal amb el qual interactuem amb la màquina remota com si fóssim aquí mateix, és a dir, amb accés a tots els recursos (sempre que tinguem els permisos corresponents). Això sí: aquest tipus d'accés és molt més avançat i requereix coneixements sobre el sistema operatiu Linux o UNIX.

4.5. Gestors de bases de dades

Encara que alguns CMS no necessiten una base de dades (usen arxius de text o XML com a sistema per a emmagatzemar la informació), la majoria sí que en necessiten.

En la base de dades del CMS no sols es guarda el contingut del lloc web, sinó també la configuració i qualsevol dada que s'hagi d'emmagatzemar per a utilitzar-la més endavant. Per tant, si volem traslladar el nostre lloc web des del servidor de proves al de producció, hem de tenir en compte quins valors de la configuració es guarden en la base de dades per a preveure problemes.

El gestor de bases de dades més utilitzat en CMS de codi obert és el **MySQL**, una **base de dades relacional** basada en **SQL**, propietat de Sun Microsystems des de 2008 i d'Oracle des del 2009 (quan Oracle va adquirir Sun) i que està instal·lat en milions de servidors a tot el món. La popularitat del MySQL està molt lligada a PHP, encara que és un servidor totalment independent de PHP i el pot fer servir pràcticament qualsevol llenguatge de programació.

El MySQL està basat en SQL, un llenguatge estàndard d'accés a bases de dades relacionals. Les **bases de dades relacionals** estan formades per taules que contenen registres (fitxes d'informació) que, al seu torn, contenen camps. Es diuen *bases de dades relacionals* perquè els camps de les diferents taules es poden relacionar.

Per exemple, podem tenir un camp autor apuntant a una taula d'autors.

L'**SQL** és un llenguatge d'alt nivell en què les instruccions permeten crear, llegir o modificar els registres de la base de dades mitjançant instruccions com *select*, *insert*, etc.

En principi, és possible accedir a diferents servidors de bases de dades amb la mateixa sintaxi de l'SQL, encara que hi ha variants que poden limitar la compatibilitat entre sistemes. L'SQL és un llenguatge basat en "línia d'ordres" (instruccions de text que s'executen en un terminal), però hi ha utilitats, com PhpMyAdmin (basat en PHP), que permeten un accés molt més visual i intuïtiu al servidor MySQL. Altres gestors de bases de dades basats en SQL són SQLServer (Microsoft) o PostgreSQL.

Sqlite (http://www.sqlite.org/) és un cas especial, ja que és una biblioteca de programació que permet fer consultes SQL estàndard sense servidor de base de dades, i totes les dades es desen en un únic arxiu. Això té l'avantatge que es pot usar, per exemple, en aplicacions de mòbil per a Android o iOS.

4.6. Llenguatges de programació

A més d'usar una base de dades, tots els CMS estan implementats en algun llenguatge de programació web. El **llenguatge utilitzat condiciona el servidor** on finalment tindrem executant la nostra aplicació quan estigui en explotació. En la fase de definició tècnica sempre haurem de triar un CMS que s'adapti als requisits del servidor on el volem instal·lar.

Els llenguatges de programació web més usats actualment són els següents:

- **Active Server Pages** (ASP, ASP.net). Solució propietària de Microsoft.
- **Java**. De Sun Microsystems (actualment, Oracle).
- **PHP**. Llenguatge dissenyat inicialment per a crear pàgines dinàmiques, el més usat en els CMS actuals.
- **Ruby**. Un dels més recents (va aparèixer el 1995). Sovint s'usa Ruby en un *framework* de programació anomenat *Rails* (Ruby on Rails).
- **Python**. Llenguatge de sintaxi simple i senzilla.

Curiositats

El llenguatge Ruby (robí) està inspirat en contraposició al nom d'un altre llenguatge, Perl (perla). D'altra banda, el nom del llenguatge de programació Python està, efectivament, inspirat en el grup d'humor britànic Monty Python.

Observació

Per la seva naturalesa, la immensa majoria de llenguatges de programació web són interpretats. Això garanteix que les aplicacions puguin funcionar en múltiples plataformes. A més, molts també són "orientats a objectes" i de codi lliure.

4.7. Allotjament web (*hosting-housing*)

Una qüestió important en qualsevol aplicació web és la de l'allotjament. Durant el desenvolupament, normalment, i en la fase d'explotació pública, el projecte estarà funcionant en un servidor extern contractat a una empresa ex-

terna. Encara que tècnicament és possible tenir el servidor web en la nostra empresa, hi ha diversos motius pels quals és molt millor contractar un servei extern:

- Els *data centers* (com es diuen les instal·lacions on s'allotgen els servidors de les empreses d'allotjament) estan preparats per a mantenir una temperatura òptima per al funcionament dels servidors i per a fer front a possibles talls del subministrament elèctric.

- Disposen de personal preparat en cas d'incidències.

- Disposen de bons canals de comunicació amb gran amplada de banda.

- Optimitzen recursos en compartir-los entre diversos clients.

Aquestes raons expliquen per què hi ha molt poques empreses que allotgin elles mateixes la seva pàgina web.

Un allotjament pot ser bàsicament de tres tipus, en aquest cas organitzats de menys a més costós:

- **Servidor compartit**. En aquest cas, diversos llocs web són allotjats en el mateix servidor físic. És una opció menys costosa però no té el rendiment necessari en projectes amb un gran nombre de visites. A més, no tenim accés a la configuració bàsica de la màquina i, per tant, no podem instal·lar aplicacions que no vinguin per defecte.

- **Servidor virtual privat**. Es tracta d'una màquina virtual que funciona com un servidor privat complet i que s'executa com a tal. Té els avantatges d'un servidor dedicat però amb un cost més baix; el rendiment no és tan alt com en el cas del servidor dedicat.

- **Servidor dedicat**. Consisteix a llogar una màquina física completa, de manera que es pot configurar a mida. Es tracta de l'opció més cara, que només s'utilitza en llocs amb un gran nombre de visites o necessitats molt especials.

Normalment, tenim l'opció de començar amb un servidor compartit i, a mesura que el nombre de visites augmenta (i, per tant, el trànsit), passar després a un servidor dedicat. Si el nombre de visites continua augmentant, haurem de passar d'un a diversos servidors dedicats. Aquí és on el nostre CMS ha de tenir la possibilitat d'"escalar" bé en un entorn d'aquest tipus (amb diversos servidors simultanis).

Per descomptat, llocs web amb un gran nombre de visites, com flickr.com, facebook.com, etc., no solament tenen diversos servidors dedicats sinó que fins i tot disposen d'un o més *data centers*.

Facebook, per exemple, disposa de més de 60.000 servidors per al seu funcionament (dades de 2012).

4.8. Manteniment, còpies de seguretat

Un problema important en un sistema informàtic és fer-ne correctament el manteniment. Periòdicament cal comprovar si hi ha **actualitzacions** del sistema o d'extensions o temes que es fan servir. Si cal actualitzar arxius, és important fer una còpia de seguretat per a poder recuperar les dades en cas que alguna cosa falli.

Normalment, el mateix CMS disposa de mecanismes d'avís quan algun mòdul o component necessita una actualització. En aquest cas, haurem de seguir les instruccions.

En alguns casos, l'actualització és gairebé automàtica (com en WordPress), però en altres haurem de descarregar nosaltres els arxius, pujar-los en la ruta correcta i seguir algun pas documentat en el CMS concret que estiguem utilitzant.

És molt important actualitzar el nostre sistema, ja que en cas contrari possiblement deixarem oberts forats de seguretat pels quals poden entrar *hackers* amb males intencions, o simplement per a demostrar que ho poden fer, o programes indesitjats que injectin *spam*.

En general, a part de les actualitzacions, sempre hem de fer **còpies de seguretat** (o ***backups***) periòdiques, especialment de la base de dades, ja que aquí és on es desen tant les dades dels continguts com la configuració i personalització que hàgim fet en el sistema.

El més pràctic és disposar d'alguna eina de còpies de seguretat automàtiques. Alguns serveis d'allotjament (*hosting*) disposen d'aquestes eines, que permeten indicar fàcilment amb quina periodicitat volem fer una còpia de seguretat de la base de dades per a poder-la recuperar en cas de problemes.

4.9. Servidor de proves local

Sovint és interessant usar un servidor local de proves per al desenvolupament. Una vegada hàgim completat el projecte en local, pujarem els arxius al servidor definitiu (copiant els arxius i la base de dades i ajustant la configuració).

Observació

És el que en argot informàtic es denomina un *deploy* o *deployment*: passar del sistema temporal o de proves a l'operacional o d'explotació.

El fet d'usar un servidor local durant el desenvolupament té diversos avantatges:

- És molt més eficient i ràpid, podem editar els arxius en local (no necessitem accedir a Internet), per la qual cosa podem provar qualsevol canvi amb més rapidesa.

- El nostre projecte no serà públic (accessible des d'Internet) mentre l'estem desenvolupant i, per tant, serà més segur si volem mantenir la privacitat del projecte.

- Si volem, podrem visualitzar el lloc web des de la xarxa local de la nostra organització. Si el servidor forma part de la xarxa local, podrem accedir des de qualsevol ordinador de la xarxa usant adreces IP locals (com, per exemple, "192.168.1.10").

A efectes pràctics, no hi ha cap diferència entre un servidor local (en la nostra màquina) i un servidor remot, a part dels punts anteriors.

Si usem una plataforma LAMP, hi ha diversos paquets que integren el servidor web Apache juntament amb PHP i MySQL per a poder-los usar en mode local; alguns d'ells són:

- **Xampp** (http://www.apachefriends.org/es/xampp.html) (vàlid tant per a Mac com per a Windows i Linux).

- **EasyPHP** (http://www.easyphp.org/) (només Windows).

- **Wamp** (http://www.wampserver.com/en/) (només Windows).

- **Mamp** (http://www.mamp.info/en/index.html) (només Mac).

En alguns sistemes operatius, com Mac OS o Linux, aquests servidors ja estan instal·lats per defecte (i, per tant, n'hi ha prou de seguir les instruccions del sistema operatiu en qüestió).

Quan treballem en local en comptes de treballar amb una adreça "pública", el nostre servidor té una adreça privada només accessible des del nostre ordinador, normalment amb l'adreça "http://localhost" o "http://127.0.0.1".

Observació

Per a provar si tenim algun servidor local funcionant en el nostre sistema, s'ha d'entrar en un navegador web l'adreça *localhost* i veure què apareix. Si apareix un "error 404", en aquest moment no tenim cap servidor local funcionant. En cas contrari, observeu de quin servidor es tracta.

El servidor web sempre utilitza un directori del nostre sistema d'arxius (normalment anomenat *www* o *htdocs*) en què hi ha d'haver els arxius per a poder accedir pel navegador.

Així, per exemple, en copiar un arxiu "test.html" dins d'una carpeta "test" dins de "www" o "htdocs", podrem accedir usant el navegador en l'adreça http://localhost/test/test.html.

A més, tindrem accés a altres utilitats com el gestor de la base de dades (normalment phpMyAdmin, si utilitzem MySQL). Depenent del sistema escollit, probablement aquesta opció apareixerà en el menú de l'eina.

Usant un servidor local podrem instal·lar i provar qualsevol gestor de continguts ràpidament.

Exercici

Descarregar un d'aquests servidors locals (per exemple Xampp, o usar el que hi ha instal·lat per defecte, si s'escau), instal·lar-lo, activar-lo i accedir mitjançant el navegador a l'adreça localhost per a comprovar que funciona correctament. Buscar l'accés a phpMyAdmin i veure si ja hi ha bases de dades de prova definides.

5. Característiques d'un bon CMS

Hi ha una sèrie de característiques que qualsevol bon gestor de continguts o CMS ha de complir. L'avantatge principal d'un CMS és que aquestes característiques **s'activen automàticament**, pel simple fet d'usar-lo, sense esforç addicional per part de l'editor, administrador o desenvolupador de l'aplicació.

5.1. Separació de contingut i presentació

La **separació de contingut i presentació** és possiblement la característica més important. El CMS s'ha d'encarregar de gestionar els continguts i l'estructura, però el nivell de presentació ha d'estar totalment separat, normalment en forma de **temes** (*themes*).

De la mateixa manera que usant HTML i CSS separem el contingut de la presentació, en un CMS complim el mateix objectiu utilitzant temes.

Un **tema** està construït per unes **plantilles** en algun llenguatge de programació (com PHP) i un o diversos arxius CSS vinculats.

El fet d'utilitzar temes té múltiples avantatges:

- Assegurar la validació correcta de l'HTML i el CSS, encara que modifiquem el contingut, ja que la validació només depèn de la validació del tema que estiguem usant.
- Poder visualitzar correctament el contingut en diferents plataformes, com les mòbils (només haurem d'activar un tema diferent en aquest cas).
- Assegurar-nos que l'editor del contingut no modificarà el codi HTML o CSS, impossibilitant trencar la validació d'HTML i CSS.
- Poder canviar el disseny del nostre lloc en qualsevol moment i per qualsevol causa, sense haver de modificar els continguts per a res.

Observació

Aquests avantatges només hi són en el cas que l'editor integrat en el CMS no permeti la introducció de codi HTML, que podria alterar la validació i presentació de les pàgines.

En un CMS sempre tenim la possibilitat de descarregar i activar temes desenvolupats per la comunitat, però també la de crear els nostres segons les necessitats que tinguem.

5.2. Facilitat d'ús del panell d'administració

Un aspecte crucial en qualsevol CMS és la facilitat d'ús del **panell d'administració**. Encara que el sistema tingui moltes funcionalitats avançades, no ens servirà de res si aquestes no són fàcils d'utilitzar i de comprendre.

Hem de tenir en compte que el panell d'administració normalment té, almenys, **dos tipus d'usuaris**, cadascun dels quals entrarà amb el seu compte especial al panell i tindrà permisos per a fer algunes tasques, però no per a fer-ne unes altres:

- **Administrador**. Les opcions enfocades a l'administrador del lloc s'han de configurar en el moment de desenvolupar, però normalment després no es modificaran. Aquest seria el cas d'opcions avançades com la configuració de la memòria cau (*cache*), sobre si es publiquen per defecte els continguts en la portada o no, blogs en la pantalla, construcció del lloc en general, etc.

- **Editor**. Les opcions que apareixen per a aquest perfil són les referents al contingut i les que s'usen mentre s'estan actualitzant i mantenint el contingut durant la vida útil del lloc, sense afectar ni el funcionament ni l'estructura del lloc.

En els CMS més avançats és possible definir tants perfils com vulguem, cadascun amb permisos específics.

D'aquesta manera, podríem definir, per exemple, un tipus d'usuari "traductor" que té permís per a traduir continguts però no per a publicar-los.

Des d'aquest punt de vista de la usabilitat, per a aquests usuaris un panell d'administració ha de tenir les característiques següents:

- Ser **intuïtiu**. Les opcions han de ser autodescriptives. Idealment, no hauríem de necessitar un manual per a poder començar a usar-lo (encara que sempre hi haurà una corba d'aprenentatge).

- Proporcionar un **editor WYSIWYG** (*what you see is what you get*), de manera que no es necessitin coneixements bàsics d'HTML i, més important, evitar errors pel tancament incorrecte d'etiquetes HTML, per exemple, o la inclusió d'elements que podrien modificar el disseny. En aquests editors és convenient poder activar l'ús de determinades etiquetes (com capçaleres, llistes, *strong* o *em*), però no d'altres que podrien afectar la presentació de l'HTML.

- Permetre la **previsualització**. Opció de poder previsualitzar el contingut abans de publicar-lo definitivament, en la plantilla (tema) que tinguem activada, de manera que puguem visualitzar el contingut tal com el veurà l'usuari final. En alguns CMS el panell d'administració té un aspecte dife-

rent del lloc final (és el cas de WordPress), per la qual cosa això és especialment important.

5.3. Seguretat

En qualsevol sistema informàtic complex (com és el cas que estem veient) hi ha la possibilitat que hi hagi **forats de seguretat** o **vulnerabilitats** que siguin aprofitats per gent i programes maliciosos.

Qualsevol forat de seguretat podria permetre que persones o programes automàtics (*bots*), entre altres coses:

- Extreguin informació privada dels comptes dels usuaris del lloc.
- Publiquin continguts directament sense permís (*spam*).
- Utilitzin l'allotjament web per a altres propòsits no desitjats.

En general, quan es detecta un forat de seguretat, la comunitat de desenvolupadors del CMS soluciona ràpidament el problema oferint una actualització del programari. Per això és important actualitzar els arxius del nostre CMS periòdicament, per a evitar al màxim que aquests problemes de seguretat afectin el nostre lloc.

Cal destacar que, encara que hi pot haver aquests problemes, normalment els CMS són molt més segurs que un sistema fet a mida, ja que en aquest cas no hi ha tants recursos per a poder revisar contínuament el codi i detectar vulnerabilitats. D'altra banda, també és cert que els atacs són menys probables perquè les vulnerabilitats són menys conegudes. És a dir, la seguretat superior d'un CMS està acompanyada d'una difusió més gran en cas que hi hagi un error de seguretat. Així, quan en un CMS àmpliament utilitzat hi ha una vulnerabilitat, aquesta vulnerabilitat es coneix ràpidament, tant per part de la gent amb bones intencions com de la gent amb possibles mals propòsits. Motiu de més per a estar alerta i instal·lar les actualitzacions sempre que apareguin.

5.3.1. Algunes amenaces a la seguretat

Hi ha algunes tècniques malicioses que permeten entrar en sistemes informàtics i respecte a les quals els CMS han d'estar protegits. Algunes de les més conegudes són:

- **Cross side scripting** (o **XSS**). Es tracta d'executar instruccions en la part del client del navegador (mitjançant JavaScript o Flash, per exemple) des de llocs remots simulant que s'executen en el mateix domini, amb l'objectiu de tenir accés a informació del nostre lloc o modificar informació, falsificar galetes (*cookies*) per a simular sessions d'usuaris ficticis, etc.

- **SQL injection**. Aquesta tècnica consisteix a "injectar" instruccions d'SQL en les variables de l'URL amb l'objectiu de tenir accés a la base de dades del lloc. Per exemple, passar en l'URL com a variable alguna cadena de l'estil de `"?q=delete from usuaris.."` podria tenir efectes directes sobre la base de dades si el codi no està protegit contra això.

En tots dos casos la solució és en el manteniment per part dels desenvolupadors i l'actualització per part dels usuaris del codi de l'aplicació amb l'objectiu de protegir-se d'aquests possibles atacs. Es tracta d'un procés continu en el temps, ja que sempre és possible que es detectin nous problemes que no s'havien detectat anteriorment. A la protecció d'aquest tipus de vulnerabilitats corresponen la majoria de les actualitzacions que s'han de fer en un CMS periòdicament.

Una altra vulnerabilitat diferent és l'***spam***. Es denomina *spam* qualsevol missatge o comentari no sol·licitat, normalment amb intencions publicitàries. Aquests comentaris són enviats per programes automàtics que rastregen Internet cercant llocs on poder publicar.

És recomanable usar algun sistema de protecció contra l'*spam*, com per exemple els **CAPTCHA**. Un CAPTCHA és un sistema desafiament-resposta que té com a objectiu determinar si qui publica el contingut o comentari és una persona o un robot (programa informàtic preparat per a això). Normalment, el desafiament s'implementa en forma de lectura d'una imatge amb paraules distorsionades (perquè un reconeixedor òptic de caràcters o ROC, OCR en anglès, no ho pugui interpretar) o d'una operació matemàtica. D'aquesta manera els *bots* no poden publicar informació, ja que no són capaços de desxifrar el contingut de la imatge.

Normalment, l'opció d'afegir CAPTCHA en un CMS es proporciona per mitjà d'extensions (*plug-ins* o mòduls, segons la denominació de cada CMS).

5.4. Extensibilitat (*plug-ins* o mòduls)

Definim l'extensibilitat com la capacitat d'augmentar les funcionalitats del sistema amb el mínim cost de desenvolupament possible. En la majoria de CMS això s'aconsegueix instal·lant extensions, **mòduls** o ***plug-ins*** (la nomenclatura varia segons el CMS però el concepte és el mateix).

En els CMS més utilitzats (com WordPress o Drupal), el nombre d'extensions es compta per milers i en permet ampliar enormement el nombre de funcionalitats. Aquestes extensions solen ser desenvolupades i mantingudes per individus o equips de persones que normalment no pertanyen a l'equip de desenvolupament del CMS principal.

En qualsevol cas, abans d'usar una extensió, sempre cal assegurar-se que la versió implementada és **estable** (sense errors importants detectats). Una bona manera de saber si una extensió és prou estable és que la base d'usuaris sigui important.

Aquestes ampliacions se solen programar a partir d'una plataforma pròpia del mateix CMS (API o *framework*), de manera que es garanteix una capa de separació del nucli del sistema amb el de la funcionalitat que volem proporcionar (també per a evitar problemes de seguretat). Usant API, hi ha la possibilitat de crear les nostres pròpies ampliacions en cas que no en trobem cap que s'adapti a les nostres necessitats.

Quan instal·lem una extensió, sempre ens hem d'assegurar que pertany a la mateixa versió que el nucli del CMS i que està prou provada per la comunitat (en cas contrari, podem tenir problemes). També és aconsellable fer sempre una còpia de seguretat de la base de dades abans d'instal·lar-la.

Observació

Des del punt de vista del desenvolupador: sempre que vulguem crear una nova funcionalitat que no existeix en alguna extensió de la comunitat, l'haurem de desenvolupar, mai no hem de modificar el codi base del CMS (el que es denomina en argot "hackejar" el codi).

5.5. Escalabilitat

L'escalabilitat d'un sistema informàtic es defineix com la "capacitat de mantenir la capacitat del servei a mesura que la càrrega sobre el sistema augmenta".

La **càrrega sobre el sistema** es mesura en nombre de visites simultànies. Les visites es poden mesurar de diverses maneres:

- Nombre de peticions (*hits*). És a dir, el nombre d'arxius que el servidor ha hagut de proporcionar (incloent HTML, CSS, imatges, etc.).

- Nombre de pàgines vistes.

- Nombre de visites úniques. Aquest és un dels paràmetres més utilitzats per a mesurar l'audiència d'un lloc web.

Tots aquests paràmetres es mesuren per unitat de temps: per exemple, nombre de visites per dia o pàgines vistes per mes. L'escalabilitat, com dèiem, és la capacitat que té el sistema per a assumir un nombre de visitants, és a dir de trànsit i volum de dades, cada vegada més alt sense tenir problemes si el sistema està ben dissenyat, tindrà mecanismes que n'assegurin el funcionament a mesura que pugi el nombre de visitants.

Un bon CMS ha de permetre adaptar-se al creixement de visites sense plantejar problemes, encara que en realitat, en gran part, l'escalabilitat s'ha d'implementar en el servidor (augmentant el nombre de servidors dedicats, replicant la base de dades, etc.).

El CMS ha d'aportar mecanismes que ajudin a l'escalabilitat, com la ***caché*** o el ***throttling***.

- ***Caché***. Alguns CMS emmagatzemen les pàgines en la base de dades en comptes de generar-les cada vegada mitjançant PHP i accés a la base de dades. D'aquesta manera es pot incrementar enormement el rendiment i assumir un nombre de visites molt més gran. No és el mateix haver de fer vint peticions a la base de dades que només una (recuperar la pàgina ja construïda de la base de dades o de la memòria).

- ***Throttle***. És la capacitat d'alguns CMS de desactivar algunes funcionalitats (no bàsiques) en el moment en què es detecta un augment important de peticions al lloc web, i d'aquesta manera garantir que la resta del sistema continua funcionant.

L'escalabilitat és una especialitat de la gestió de servidors complexa en si mateixa que han de dur a terme especialistes. A mesura que les visites augmenten considerablement (en xifres de diversos milers de visitants al dia), cal replicar servidors i bases de dades, optimitzar la *caché*, etc. Si estan ben implementats, aquests passos són independents de l'ús d'un CMS o altre.

Una garantia que la nostra elecció permet una bona escalabilitat és observar si hi ha llocs amb un gran nombre de visites que usin el nostre CMS. És el cas de WordPress, Drupal, openCms i altres dels més utilitzats.

5.6. Estàndards

Igual que qualsevol altre lloc web, una aplicació desenvolupada amb un CMS hauria de complir els **estàndards** actuals respecte a l'HTML i el CSS. Complir aquests estàndards proporciona diversos avantatges:

- El codi HTML pot ser més ordenat i semàntic.

- Millora el manteniment futur. És més fàcil d'actualitzar i probablement funcionarà millor en navegadors que encara no han aparegut.

- Faciliten que les pàgines es vegin bé en qualsevol navegador.

- És una condició necessària per a fer el contingut accessible.

- Millora el posicionament en cercadors.

La manera de saber si un lloc web compleix amb els estàndards web és passar per un **procés de validació**.

Observació

El W3C ofereix un servei de validació per a HTML a http://validator.w3.org/ i de CSS a http://jigsaw.w3.org/css-validator/. Per a comprovar que el nostre CMS valida correctament només cal verificar-ho amb aquestes eines.

A causa que la presentació i el contingut estan separats, la validació depèn en bona mesura de la capa de presentació, és a dir, del tema actiu en aquest moment. És responsabilitat del creador del tema que el codi HTML/CSS sigui vàlid.

Per a garantir que el nostre projecte compleix els estàndards web ens hem d'assegurar dels punts següents:

- El codi HTML/CSS de la plantilla que hem escollit o desenvolupat és estàndard.
- L'editor no permet introduir codi HTML no vàlid (la qual cosa trencaria la validació).
- Formació en estàndards web del personal que s'encarrega de l'actualització. Aquest és un tema molt important, ja que, si no, el lloc web podria deixar de ser vàlid en qualsevol moment.

5.7. Sindicació de continguts (RSS i Atom)

A més de la manera tradicional de veure continguts a Internet, això és, visitant pàgines web o utilitzant un cercador per a trobar continguts rellevants, hi ha sistemes amb els quals ens podem subscriure a les fonts d'informació. D'aquesta manera, la informació ve cap a nosaltres en lloc d'haver de dedicar temps a explorar i a navegar per a saber si hi ha nous continguts en les pàgines que visitem habitualment.

Així, en subscriure'ns a una font d'informació mitjançant un **lector de fonts d'informació** (també anomenat *lector* o *agregador de feeds* o RSS), l'aplicació ens avisa quan hi ha algun contingut nou que encara no hem visitat, i fins i tot podem llegir el contingut d'aquesta font directament en l'aplicació sense haver de visitar el lloc original.

Alguns d'aquests lectors de *feeds* funcionen en línia, com Google Reader.

Per a poder-nos subscriure a aquestes fonts d'informació necessitem que aquestes publiquin (**sindiquin**), a més de les pàgines convencionals HTML, uns arxius en un format estàndard. Aquests formats estàndard són els denominats **RSS** i **Atom**. Es tracta d'arxius de text en format XML amb etiquetes especials específiques que inclouen tot el contingut més recent que s'ha publicat en el

nostre gestor de continguts i que serveixen per a delimitar el contingut de tots els articles del lloc. Tant els formats RSS 1.0 i 2.0 com Atom compleixen la mateixa funció.

Atès que la **sindicació de continguts** és una cosa necessària avui dia, hauria de ser una de les característiques obligatòries en qualsevol CMS.

Si un CMS està ben dissenyat, en publicar continguts en el panell d'administració aquests també s'haurien de publicar automàticament mitjançant RSS sense haver-nos de preocupar de cap acció específica per a això.

La sindicació RSS no és útil solament en les plataformes de blogs (com WordPress) sinó que també ho és en qualsevol lloc web que tingui continguts que s'actualitzen periòdicament.

El fet de publicar automàticament en RSS o Atom té múltiples avantatges:

- Els usuaris es poden subscriure als continguts usant un lector de *feeds*, eina que, com acabem de dir, permet visualitzar automàticament el contingut nou que es genera en les fonts d'informació a les quals s'està subscrit.

- Millora la indexació en els cercadors (especialment en els cercadors específics de blogs com *Google Blog Search*).

- Permet que la gent agregui la informació del nostre blog en altres llocs web. Per exemple, creant una pàgina personalitzada en serveis com Netvibes o similars, o en el seu propi blog o pàgina web.

Per descomptat que la sindicació de continguts hauria de ser opcional i configurable, podent decidir, per exemple, sobre quins tipus de continguts es fa i sobre quins no, o fins i tot quins camps del contingut es publiquen (fotografies o altres dades). L'habitual és que tinguem l'opció de sindicar el contingut en format resumit (obligant el lector a visitar la font original) o de publicar tot el text en el RSS.

5.8. Accessibilitat

Una altra de les bones característiques que hauria de tenir qualsevol CMS és un grau alt d'accessibilitat. Perquè qualsevol persona en qualsevol moment pugui utilitzar el nostre lloc web, hi ha **mecanismes reconeguts i estandarditzats internacionalment que permeten que el contingut sigui accessible** mitjançant lectors especialitzats.

De la mateixa manera que en la validació HTML/CSS, l'accessibilitat depèn en gran mesura de la implementació d'un tema en concret que estiguem usant en el nostre CMS. Perquè un tema sigui accessible és un requisit necessari, encara que no suficient, que validi correctament.

També és important que tots els continguts estiguin ben etiquetats perquè els lectors de pantalla o altres dispositius els puguin entendre. En un document web ben marcat semànticament sempre resulta més fàcil de navegar i d'interpretar per a qualsevol dispositiu.

En general, una bona manera de pensar en l'accessibilitat és no donar per garantit que els usuaris navegaran usant un ratolí, ni que podran veure les imatges. Sempre és bona idea oferir continguts alternatius si fem servir tecnologies com Flash, etc.

Tota la informació sobre els criteris estàndard d'accessibilitat es poden trobar en la pàgina del W3C. Trobarem les recomanacions WAI (*web accessibility initiative*), encara que aquestes directrius són del 1999 i han quedat una mica obsoletes. Les WCAG 1.0. es basen en catorze directrius que permeten donar una classificació de conformitat (A, si es compleixen les de prioritat 1; AA, les de nivell 2, i AAA, la classificació més elevada). Les WCAG 2.0 (http://www.w3.org/TR/WCAG20/), més evolucionades, no depenen tant de la tecnologia, per la qual cosa és difícil trobar un sistema automàtic per a comprovar-ho.

Un dels components que poden causar problemes d'accessibilitat en el nostre CMS són les extensions o mòduls "contribuïts" (és a dir, desenvolupats per gent externa a l'equip que manté el nucli del CMS) si no estan desenvolupats tenint en compte l'accessibilitat. L'única solució, a més de llegir bé la documentació, és provar-los i veure si compleixen els estàndards.

Desgraciadament, no hi ha mètodes automàtics per a comprovar íntegrament que el nostre lloc web és accessible, encara que es poden fer tests parcials mitjançant programes i la resta s'ha de comprovar manualment fent proves amb usuaris reals.

Igual que en el cas dels estàndards web, només podrem garantir l'accessibilitat mitjançant tres factors:

- El CMS i el tema que estiguem utilitzant ha de ser accessible en la mesura del possible.
- L'editor ha de permetre introduir contingut sense perjudicar-ne l'accessibilitat.

Pàgina web

Hi ha algunes eines de comprovació automàtica de l'accessibilitat:

http://www.rnib.org.uk/professionals/webaccessibility/testingtips/Pages/automated_testing.aspx.

- Per a assegurar-nos que el lloc continua essent accessible en el temps, cal preveure un pla de formació sobre accessibilitat del personal que manté el lloc web.

Plone, un dels CMS que veurem més endavant, és un dels que compleix millor els criteris d'accessibilitat, encara que la resta també sol tenir en compte aquests criteris i sovint es poden completar amb la instal·lació de mòduls addicionals.

5.9. Estabilitat

En informàtica se sol definir com un **sistema estable** el que és tolerant a fallades o errors, és a dir, el que no generarà ni produirà errors si s'utilitza correctament.

Per a assegurar-nos que el nostre CMS és estable ens hem de fixar en els punts següents:

- El CMS que hem triat té una base d'usuaris i desenvolupadors prou gran? En cas contrari és difícil que s'hagi verificat encara a fons en diverses situacions.

- Estem fent servir una versió etiquetada com a estable? No una versió beta o alfa que encara no està comprovada totalment.

- El servidor que estem fent servir compleix els requisits mínims d'instal·lació? Si, per exemple, la versió del llenguatge de servidor que utilitzem no compleix amb els requisits, és possible que el sistema funcioni però amb errors.

5.10. Facilitat d'instal·lació i manteniment

Un factor essencial perquè un CMS sigui àmpliament utilitzat per la comunitat és la facilitat d'instal·lació. És per això que molts CMS són extremadament senzills d'instal·lar en un servidor (sempre que sapiguem les dades essencials). A més, aquestes facilitats s'han d'estendre també al manteniment posterior. Les **actualitzacions** del programa o **extensions** associades han de ser fàcils d'instal·lar. Qualsevol CMS actualitza el seu codi al llarg del temps (per exemple, per solucionar errors o tapar forats de seguretat), per la qual cosa també és necessari oferir algun sistema d'actualització, que idealment hauria de ser el més automàtic possible. El mateix s'aplica a les possibles extensions que tingui el sistema.

En aquest sentit, un dels millors exemples és el WordPress, en què la instal·lació d'extensions i l'actualització del sistema es fan directament des del panell d'administrador sense haver d'accedir en cap moment al sistema d'arxius o haver de buscar per Internet les actualitzacions, descarregar o descomprimir arxius, etc. En aquest CMS s'ha fet un gran esforç per a facilitar al màxim el treball de l'administrador del lloc, cosa que ha repercutit sens dubte en el fet que sigui una de les plataformes actualment més utilitzades.

Els creadors de CMS saben que un sistema complex d'instal·lar és una barrera d'entrada a usuaris amb pocs coneixements informàtics i, a més, pot ser una font d'errors i de problemes.

Idealment, un CMS s'hauria de poder instal·lar sense haver d'editar a mà cap arxiu de configuració, encara que sempre és important tenir accés a totes les opcions de configuració i, si es vol, poder-les editar a mà.

5.11. Facilitat d'adaptació i personalització

L'adaptació i personalització dels CMS es pot fer, gairebé sempre, en dos dels seus aspectes bàsics:

- **Funcionalitats**. Incorporant extensions, que permeten modificar-les o augmentar-les.

- **Aspecte**. Mitjançant **temes** que permeten personalitzar la presentació.

Un bon CMS no solament ha de tenir la possibilitat d'afegir extensions i temes, sinó que també ha de disposar de recursos perquè, al seu torn, qualsevol desenvolupador els pugui crear i, idealment, deixar-los després a la disposició de la comunitat. A més, un bon CMS hauria de disposar d'**API** de programació que permetin l'accés a les funcions principals del CMS per a poder-les modificar si cal. Es tracta de poder treure profit a les funcionalitats del CMS i al mateix temps poder crear funcionalitats noves no previstes.

Aquest és el cas del Codex (http://codex.wordpress.org/) de WordPress o de la documentació de l'API de Drupal. En aquestes pàgines estan documentades totes les funcions que usen aquests dos sistemes.

Quan s'efectua una personalització, sempre és molt important modificar només els arxius del mòdul o tema en concret, no modificar mai els arxius base del CMS, ja que si es fa una actualització es perdrien els canvis i a més es podria afectar la seguretat i la integritat del sistema.

5.12. Interconnectivitat

La **interconnectivitat** es defineix com la capacitat d'un CMS de connectar-se a altres sistemes de l'organització (per exemple, a bases de dades existents). És una característica molt important en sistemes empresarials (igual que en altres sistemes informàtics), ja que permet:

- Compartir recursos.
- Tenir accés instantani a bases de dades compartides.
- Administrar la xarxa de manera centralitzada.
- Reduir pressupostos (temps, diners).
- Optimitzar recursos humans.

Un cas típic seria **LDAP**, un sistema estàndard d'autenticació de recursos usat per moltes organitzacions. Integrar LDAP en el nostre CMS permet que els usuaris s'autentiquin en LDAP, sense necessitat de crear comptes nous i validant els mateixos grups que hi pugui haver en l'organització.

Un altre cas seria l'exportació o importació de dades des de bases de dades de l'organització al CMS o al revés.

> Per exemple, imaginem una empresa que ja disposa d'un estoc de productes amb referències i preus en una base de dades interna. Si aquesta empresa vol obrir una tenda a Internet, el lògic seria que la pàgina web utilitzés aquesta mateixa informació en lloc d'haver-la de duplicar.

Les aplicacions web que requereixen interconnectivitat són complexes, ja que no depenen només del CMS, sinó també de sistemes externs que poden ser molt variats, amb tecnologies diferents i formats que no sempre són fàcils d'integrar en el nostre CMS. Moltes vegades caldrà programació a mida per a poder aconseguir l'objectiu.

5.13. Possibles aplicacions integrades en un CMS

En els CMS més generalistes és important veure quines **aplicacions** (**funcionalitats**) es poden implementar de base o afegint mòduls.

Algunes de les aplicacions que poden implementar aquests CMS són:

- Blogs.
- Xats.
- Calendaris d'esdeveniments.
- PMF o preguntes més freqüents (en anglès, FAQ o *frequently asked questions*).
- *Newsletters*. Sistema automàtic per a enviar correus electrònics a subscriptors.
- Galeries de fotos.
- Cercador. Amb indexació de tot el contingut del lloc.
- *Sitemap*. Mapes del lloc en format HTML o XML.
- RSS. Sindicació de continguts.
- *Wiki*. Pàgines editables amb historial.
- *E-commerce* o comerç electrònic.

6. Altres característiques dels CMS

L'objectiu principal d'un gestor de continguts és crear pàgines estàndard en HTML i CSS a partir dels continguts de la base de dades.

Actualment, la versió més utilitzada d'HTML és la 4.01 (publicada el 1999), i la d'XHTML és la 1.0 (publicada el 2000). Quant al CSS, la versió més utilitzada és la 2.1 (2005). No obstant això, aquests estàndards estan en evolució i és important tenir-ho en compte en el gestor de continguts. A més, hi ha altres tecnologies que acompanyen a les pàgines HTML:

- **HTML5/CSS3**. Els navegadors moderns ja accepten els nous formats HTML5 i CSS3. Per a poder-los aprofitar bé, el nostre CMS ha de ser capaç d'adaptar-se.

- **JavaScript**. Avui dia la gran majoria de llocs web tenen algun tipus de programació en JavaScript, des de validació de formularis fins a refrescar dinàmicament la informació de la pàgina (AJAX). En HTML5 el paper de JavaScript es veu reforçat amb més funcionalitats. Tot sovint s'utilitza JavaScript mitjançant la biblioteca jQuery.

- **Marcació semàntica**. Hi ha diversos formats acceptats per a la marcació semàntica del contingut en pàgines web. En el futur això cobrarà cada vegada més importància. Això es pot fer amb **RDF** o **microformats**.

- **Optimització per a cercadors**. És un factor molt important; no es tracta només de presentar adequadament el contingut en formats estàndard, sinó també que els cercadors indexin fàcilment aquests continguts, de manera que puguem atreure visitants que estan interessats en la informació que publiquem.

A continuació analitzarem cadascuna d'aquestes característiques i com les hem de tenir en compte en el nostre CMS.

6.1. HTML5/CSS3

Les especificacions d'HTML i CSS continuen avançant. L'organisme internacional encarregat d'aquestes especificacions, el W3C, continua publicant noves versions, les últimes de les quals es diuen HTML5 i CSS3. El previsible és que a curt termini aquestes versions siguin les habituals en la majoria dels navegadors web, encara que en aquest moment encara no ho són del tot.

Atès que aquests estàndards són els que apliquen a la presentació d'una pàgina web, en el gestor de continguts són competència de les **plantilles** o **temes**. Qualsevol CMS pot oferir continguts en HTML5/CSS3 sempre que hi hagi una plantilla per a això o nosaltres mateixos la creem a mida. Com sempre, el problema serà mantenir la compatibilitat amb les versions anteriors dels navegadors que encara caldrà tenir en compte durant uns quants anys.

6.2. JavaScript

El **JavaScript** és un **llenguatge de programació interpretat que es pot incrustar en el codi HTML**. Tots els navegadors web tenen un intèrpret de JavaScript (tret que estigui desactivat). El JavaScript es fa servir cada vegada més, sobretot com a alternativa a Flash o altres opcions, i especialment en els nous formats web com HTML5. Fins i tot hi ha JavaScript en versió servidor (interpretat en el servidor en lloc del client).

Un bon gestor de continguts hauria de preveure com a opció la possibilitat d'incrustar codi en JavaScript. Encara que no sigui directament, hi ha multitud de ***plug-ins*** o **mòduls** en els gestors de continguts que usen JavaScript:

- En els editors de textos incrustats en la pàgina.
- En visualitzadors de fotos tipus Lightbox.
- En qualsevol efecte interactiu que no estigui usant Flash.
- Una de les possibilitats més interessants de JavaScript és sol·licitar informació al servidor sense haver de tornar a recarregar la pàgina (**AJAX**). Això permet millorar el flux d'enviament d'informació per part dels usuaris del web.

Hi ha moltes **biblioteques** de JavaScript que serveixen per a multitud de tasques, però la biblioteca més important és sens dubte **jQuery**.

6.2.1. jQuery

jQuery és una **biblioteca de JavaScript** que permet simplificar molt el desenvolupament en aquest llenguatge. Actualment, un percentatge important de pàgines web i de gestors de continguts usen jQuery (entre aquests, per exemple, el WordPress i el Drupal). Empreses importants, com Microsoft o Nokia, també la integren en els seus productes.

Bàsicament, jQuery permet:

- Abstreure el comportament dels diferents navegadors. Un dels problemes principals és que el JavaScript d'un navegador pot ser diferent del d'un altre, o variar entre les implementacions d'un navegador per a diferents

sistemes operatius. jQuery intenta separar aquestes diferències sota una capa comuna.

- Simplifica enormement les instruccions que podem usar per a aconseguir funcionalitats o efectes en nostre lloc web.

- És ideal per a desenvolupar aplicacions usant AJAX/JSON.

- Té *plug-ins* (normalment desenvolupats per la comunitat) que permeten ampliar les funcionalitats.

jQuery simplifica la manera d'interactuar amb els documents HTML, manipular l'arbre DOM, manejar esdeveniments, desenvolupar animacions i agregar interacció amb AJAX.

La primera versió és del 2006 i des de llavors s'ha establert pràcticament com un dels estàndards principals a per programar JavaScript. jQuery és programari lliure i de codi obert, amb doble llicenciament sota les llicències MIT i GNU General Public License, versió 2.

jQuery, igual que altres biblioteques, ofereix una sèrie de funcionalitats basades en JavaScript que d'una altra manera requeririen molt més codi. És a dir, amb les funcions pròpies d'aquesta biblioteca s'aconsegueixen grans resultats amb menys temps i espai.

Algunes **funcionalitats bàsiques** de jQuery són:

- Selecció d'elements DOM.

  Per exemple,

```
$('p')
```

  selecciona tots els elements *p* del document.

- Interactivitat i modificacions de l'arbre DOM, incloent suport per a CSS 1-3 i un *plug-in* bàsic de Xpath.

```
//Afegeix un element p després de tots els h2
$('h2').append("<p class='more'>More</p>");
```

- Esdeveniments.

```
$('#info').click=function( console.log("element clicat");
```

- Manipulació dinàmica del full d'estils CSS.

```
// eliminar l'estil "actiu" amb removeClass()
// i aplicar-ne un de nou amb addClass()
$(".actiu").removeClass("actiu").addClass("inactiu");
```

- Efectes i animacions.

```
//amaga l'element amb id="avís" lentament
$("#avís").hide("slow");
```

- AJAX.

```
//exemple de crida a un arxiu JSON
$.getJSON('ajax/test.json', function(data) {
```

- Diverses utilitats, com obtenir informació del navegador, operar amb objectes i *arrays*, funció trim (elimina els espais en blanc del principi i final d'una cadena de caràcters), etc.

A més, les funcionalitats de jQuery es poden ampliar mitjançant *plug-ins*.

Pàgina web

Llista de *plug-ins* de jQuery: http://plugins.jquery.com/project/Plugins.

6.2.2. AJAX/JSON

AJAX (*asynchronous JavaScript and XML* o JavaScript asíncron i XML) és una **tècnica de programació que permet refrescar part del contingut de la pàgina actual sense haver-la de recarregar**. Això s'aconsegueix amb una petició asíncrona de JavaScript al servidor, i una vegada obtingudes les dades també s'usa JavaScript per a actualitzar el contingut de la pàgina. En alguns casos, això millora molt la usabilitat de les aplicacions, i permet agilitar tasques i donar *feedback* als usuaris de manera molt més ràpida.

JSON és un format d'arxiu que emmagatzema continguts en un arbre, de manera similar a l'XML però formatat com un *array* JavaScript, de manera que es pot llegir més fàcilment. Aquest és el motiu pel qual la combinació AJAX/JSON està essent molt utilitzada.

> Un cas típic de l'ús de totes dues tecnologies és Google Maps, l'aplicació de mapes de Google. D'altra banda, gran part de les API i serveis d'Internet (com Facebook, Twitter, etc.) utilitzen JSON com a mecanisme de resposta a les crides.

Aquesta tecnologia també és molt utilitzada en CMS; per exemple, en molts casos en el panell d'administració quan tenim l'opció d'arrossegar i deixar anar (*drag-and-drop*) elements de manera visual.

Observació

Sovint la programació amb la tècnica AJAX i JSON es fa per mitjà de la biblioteca jQuery.

6.3. Suport per a plataformes mòbils

El nostre lloc web s'ha de veure correctament en diversos tipus de dispositius. La variable que afecta més en aquest sentit és la mida de la pantalla, encara que també n'hi ha d'altres com la resolució o el fet que la pantalla sigui tàctil.

Tenint en compte aquestes variables podem considerar que hi ha tres tipus principals de dispositius:

- Ordinadors de sobretaula.
- Tauletes (iPad, Android, Windows 8, etc.).
- Mòbils (telèfons intel·ligents o *smartphones*).

La responsabilitat que el lloc web es vegi correctament en tots aquests dispositius recau sobretot en el tema. Hi ha diverses tècniques (que poden funcionar simultàniament) que ha d'aplicar el tema per a abordar el problema:

- Aplicar tècniques de disseny *responsive*. Mitjançant *media queries*, en el CSS podem variar la composició i les propietats dels elements segons la mida de la pantalla actual.

- Detectar el dispositiu que està accedint al lloc i activar un tema especialment dissenyat per a cada cas (per exemple, un tema per a mòbils i un altre per a tauletes i ordinadors de taula).

- Aplicar tècniques de detecció del tipus de dispositiu en el servidor de manera que les plantilles del tema estiguin adaptades a les diferents mides de pantalla. Aquesta tècnica té l'avantatge respecte a l'opció *responsive* de poder variar realment la mida de les imatges o de no servir part dels continguts si en algun cas ho considerem necessari (cosa impossible en *responsive*, perquè és una tècnica que s'aplica en el client). Això és possible gràcies al *user-agent* que envia el navegador en les peticions. Hi ha bases de dades i biblioteques que ens permeten detectar el tipus de dispositiu a partir d'aquest *user-agent*, com wurfl (http://wurfl.sourceforge.net), i extensions i mòduls en els CMS que ens permeten modificar les plantilles del tema en funció del tipus de dispositiu que hi està accedint.

Totes les alternatives tenen les seves pròpies limitacions (si triem servir diversos temes, també triem mantenir una diversitat de temes cada vegada que modifiquem algun aspecte, per exemple). És la nostra responsabilitat avaluar les diferents opcions, amb els seus punts forts i febles, i prendre una decisió informada.

6.4. Optimització per a SEO

El **SEO** (*search engine optimization*) o **posicionament en cercadors** té gran rellevància en els gestors de contingut. Normalment, un dels objectius principals és fer arribar la informació generada amb aquests CMS als usuaris. Avui dia això s'aconsegueix en gran mesura apareixent en les pàgines de resultats per a determinades cerques.

El posicionament en cercadors és una matèria àmplia i aquest apartat no és més que un breu resum. Però, com podem optimitzar, a grans trets, el posicionament en cercadors web amb un gestor de continguts? Algunes de les condicions necessàries són:

- Tots els continguts han de tenir una pàgina única amb un títol descriptiu. Així, un portal de notícies amb mil notícies ha de tenir almenys mil pàgines úniques amb títol únic.

- Emprar codi estàndard, vàlid segons el W3C. D'aquesta manera, assegurem almenys en part que el contingut es podrà extreure fàcilment pels motors de cerca.

- És necessari que les plantilles del CMS validin i que les etiquetes CSS siguin semàntiques. El volum de continguts no hauria d'afectar aquesta validació, ja que la presentació sempre ha d'estar separada del contingut, i les plantilles sempre haurien de ser les mateixes.

- Usar adequadament els encapçalaments (h1, h2, h3, etc.).

- Afegir sempre l'etiqueta "alt" a les imatges. Especialment important per a la cerca d'imatges.

- Crear un mapa del lloc. Molts gestors de continguts permeten crear un mapa del lloc (de base o mitjançant extensions).

- L'existència d'un mapa de navegació facilita que els cercadors puguin rastrejar el contingut.

- URL amb noms "amigables" i semàntics. Tots els continguts haurien de tenir una pàgina única amb un títol que els defineixi i, si pot ser, amb un URL que també els descrigui mínimament.

  Per exemple, és molt millor "/sabata-camper-vermella" com a URL i "Sabata Camper Vermella" com a títol de la pàgina, que "index.php?=456" com a URL, i "Sabateria Muñoz" com a títol.

Algunes característiques no tan importants però també a tenir en compte són:

- Limitar el nombre de subdirectoris. Els cercadors donen més importància als continguts que estan a un nivell superior.

- Evitar enllaços inexistents. Si els usuaris generen continguts, cal assegurar-se que els URL que hi pugui haver existeixen realment; en cas contrari, pot ser motiu de penalització de posicionament pels cercadors.

- Evitar URL duplicats.

- Textos descriptius en enllaços, en lloc de "cliqueu aquí", "informe comptable 2011".

6.5. Microformats i RDF

6.5.1. Microformats

Una de les coses que semblen clares és que en el futur el web serà cada vegada més **semàntic**, o el que de vegades es denomina **web 3.0**. En aquest sentit, hi ha diverses propostes per a afegir el component semàntic al contingut HTML.

Una d'elles són els **microformats**, una proposta per a estructurar la informació i posar noms usant classes predefinides, mantenint el mateix HTML, que **serveixen per a identificar i donar valor semàntic al contingut de la pàgina HTML**.

Imaginem un exemple: en la fitxa d'un equipament municipal volem emmagatzemar la posició geogràfica. Utilitzant microformats ho podríem fer de la manera següent:

```
<span class='geo'>
   <span class='latitude'>49.205486</span>
   <span class='longitude'>-122.892036</span>
</span>
```

Pàgina web

Per saber més sobre microformats:
http://microformats.org

Aquesta informació no ha de ser necessàriament visible per l'usuari del web, ja que està pensada per a ser interpretada pels cercadors o per a ser utilitzada per extensions del navegador o per altres serveis web per a localitzar en un mapa l'equipament, emmagatzemar-lo en una agenda personal, etc.

6.5.2. RDF

Una proposta molt més avançada sobre la capa semàntica és **RDF** (*resource description framework* o marc de descripció de recursos).

En aquest cas es tracta d'afegir la **informació semàntica** seguint un estàndard definit pel W3C, que inclou, entre altres coses, la definició d'etiquetes especials en l'HTML.

Pàgines web

Per saber més sobre RDF:
http://en.wikipedia.org/wiki/Resource_Description_Framework.
http://www.w3.org/RDF/.

És desitjable que, a mesura que aquestes possibilitats semàntiques es materialitzin, els CMS les utilitzin. És el cas, per exemple, de Drupal 7, que ja implementa RDF d'origen.

7. Tipus de CMS

A continuació veurem una llista de CMS especialitzats a implementar aplicacions de diferents tipus. Començarem definint aquests tipus d'aplicacions i tot seguit veurem alguns CMS pensats per a desenvolupar projectes d'aquesta tipologia. Per a cadascun dels CMS veurem les seves funcionalitats principals, juntament amb una llista d'avantatges i desavantatges.

Els tipus d'aplicacions que hem definit són els següents:

- **genèriques**
- **fòrums**
- **blogs**
- ***wikis***
- **aprenentatge electrònic** (*e-learning*)
- **xarxes socials**
- **comerç electrònic** (*e-commerce*)

De vegades, els límits entre un tipus d'aplicació o un altre són una mica difusos. A més, cal tenir en compte que com més flexible sigui un CMS, més es podrà adaptar a diferents tipus d'aplicacions. En canvi, hi ha altres CMS tan especialitzats (com, per exemple, alguns pensats per a fòrums) que realment resulta difícil aplicar-los en altres camps.

Atès que un CMS permet bastant flexibilitat en el disseny i la composició, és gairebé impossible a primera vista descobrir sobre quina CMS s'ha construït un lloc. No obstant això, observant el codi font podem obtenir pistes que ens permetin deduir el sistema que s'ha utilitzat.

> Per exemple, en el cas de WordPress, rutes del tipus /wp- *content*/ *plugins* o /wp- *content*/ *themes*, o en el cas de Drupal, *sites*/ *all*/ *themes*.

També podem utilitzar extensions de Firefox o Chrome per a aquest propòsit (com, per exemple, Wappalyzer).

7.1. Genèriques

Una **aplicació genèrica** és la que no té una sola funcionalitat específica sinó que vol cobrir diverses funcionalitats.

A part de poder gestionar continguts dinàmics i temes, ha de complir almenys algun dels requisits següents:

- Gestió d'usuaris, rols i permisos.

- Gestió de l'estructura de continguts (tipus de continguts, camps, taxonomies).

- Possibilitat d'afegir mòduls per a cobrir aplicacions més específiques (com tendes en línia, etc.).

És a dir, els CMS d'aquest tipus estan pensats per a adaptar-se a qualsevol funcionalitat. Normalment, integren diverses d'aquestes funcionalitats i la possibilitat de crear estructures de continguts més o menys complexes, per la qual cosa les funcionalitats que no es poden implementar mitjançant les opcions bàsiques se satisfan normalment instal·lant mòduls addicionals.

El desavantatge que tenen és que, en ser més complets, el panell d'administració i la configuració també sol ser més complicada.

Casos típics d'aquest tipus de CMS són Open CMS, Joomla, Plone o Drupal. A continuació en veurem alguns amb més detall.

7.1.1. OpenCms

OpenCms (http://www.opencms.org/en/) és un gestor de continguts de codi obert desenvolupat per la companyia Alkacon Software que està implementat sobre la plataforma Java (J2EE) / XML. Java és un llenguatge semiinterpretat orientat a objectes de Sun Microsystems (avui dia, part d'Oracle).

Com en la majoria de CMS, l'administració és per web, i disposa a més d'un editor de pàgines WYSIWYG. Usa la metàfora de l'"explorador d'arxius", de manera que totes les pàgines del lloc es poden veure des de l'administrador com a arxius editables. Cadascun té camps dinàmics que es poden modificar.

Les **funcionalitats bàsiques** són:

- Panell d'administració basat en un navegador web.

- Gestió d'actius, per exemple, imatges, documents PDF, enllaços externs, etc.

- Editor de text WYSIWYG.

- Control de versions de contingut amb *roll-back* (possibilitat de recuperar versions anteriors).

- Vista prèvia de pàgines.

- Suport als continguts estructurats i no estructurats.
- Gestió integrada d'usuari i permisos del sistema.
- Plantilles basades en JSP (*Java server pages*).
- API de Java.
- Suport a la internacionalització.
- Publicació de continguts en mode estàtic i dinàmic.
- Possibilitat d'afegir mòduls.
- Funció de recerca.

Els **avantatges** principals són:

- Pensat per a funcionar sobre molts tipus de servidors web i de bases de dades diferents.
- Possibilitat de tenir pàgines estàtiques i dinàmiques.
- API en Java que permet personalitzar l'aplicació.
- Ús de JSP per a incloure elements dinàmics en la pàgina.
- Extensió mitjançant mòduls opcionals que es poden reutilitzar.
- Facilitat d'instal·lació (només les dades imprescindibles d'ubicació, base de dades i contrasenya).

I els **desavantatges**:

- És una mica complex.
- Falta d'una bona documentació i d'una bona comunitat de desenvolupadors. El fet que estigui desenvolupat per una empresa i no una comunitat és la causa d'aquest problema.
- En escriure una plantilla de JSP, els desenvolupadors han d'utilitzar l'editor de textos bàsics que es troben dins d'OpenCms. A diferència de les eines convencionals IDE com Eclipse, l'editor natiu d'OpenCms no ofereix cap comprovació de sintaxi ni ajudes sobre l'API. Tampoc hi ha depurador integrat per a ajudar en la solució de problemes complexos.

Pàgina web

Llista de mòduls i extensions per a OpenCms:
http://opencms-wiki.org/Available_Modules.

- OpenCms només proporciona una funcionalitat limitada de flux de treball entorn de contingut.

- Com en la majoria de CMS, les funcionalitats es poden ampliar instal·lant mòduls o extensions; no obstant això, el seu nombre no és tan elevat com en altres CMS.

Alguns exemples de llocs web que usen aquest CMS:

UOC. Universitat Oberta de Catalunya (http://www.uoc.edu/).

Universitat Pompeu Fabra (http://www.upf.edu/).

7.1.2. Joomla

Joomla (http://www.joomla.org/) és un CMS de codi obert amb llicència GPL (*GNU general public license*), basat en PHP/MySQL. Es tracta d'un CMS bastant flexible i adaptable a diferents necessitats.

Curiositat

El nom *Joomla* deriva de la paraula swahili *Jumla*, que significa 'tots junts'.

Joomla és una bifurcació (o *branch*) d'un altre CMS anterior, Mambo. Aquest tipus de bifurcacions és bastant comú en el camp de les aplicacions de codi obert; sense anar més lluny, Firefox és una bifurcació del codi original del navegador Netscape.

Les **funcionalitats** de la versió bàsica de Joomla (sense instal·lar extensions addicionals) són:

- Encauament de pàgines.
- Pensat per a optimitzar la indexació en cercadors.
- Genera *feeds* RSS.
- Versions per a imprimir de totes les pàgines.
- Generació de blogs.
- Fòrums.
- Enquestes.
- Calendaris.
- Cercador integrat.
- Internacionalització.

A més d'aquestes funcionalitats que hi ha per defecte, hi ha centenars d'extensions i una gran comunitat d'usuaris i desenvolupadors. Les extensions poden ser de diversos tipus:

- Components.
- Mòduls.
- Plantilles.
- *Plug-ins*.
- Llenguatges.

A més, Joomla disposa d'una API de programació i se sosté en *un framework* MVC. Per a desenvolupar components es recomana usar aquesta filosofia MVC.

Exemples de llocs desenvolupats sobre aquesta plataforma:

Museu Guggenheim (http://www.guggenheim.org/).

Oklahoma State University (http://osu.okstate.edu/welcome/).

7.1.3. Plone

Plone (http://www.plone.org/) és un sistema de gestió de continguts web també sota la llicència GPL. La seva interfície compleix els requisits d'accessibilitat WCAG-AAA i U.S. Section 508.

Plone és un CMS basat en Zope i programat en Python.

Zope i Python

Zope (http://zope.org) és un servidor d'aplicacions web de codi obert que té milers de desenvolupadors a tot el món i que està escrit en el llenguatge de programació Python. Python, per la seva banda, és un llenguatge de programació interpretat creat l'any 1991. Es compara habitualment amb altres llenguatges com Perl, Java i Ruby. Actualment, Python es desenvolupa com un projecte de codi obert, administrat per la Python Software Foundation (http://www.python.org/psf/).

És un desenvolupament basat en codi obert. Plone es pot utilitzar per a construir diversos tipus de projectes com portals, llocs web corporatius, llocs de notícies, extranets o intranets, sistemes de publicació, repositoris de documents, eines col·laboratives (*groupware*), o fins i tot per a comerç electrònic (*e-commerce*).

Altres de les possibles aplicacions de Plone és com a CRM (gestió de la relació amb els clients o *customer relationship management*) o com a sistema de gestió del coneixement (KMS, de l'anglès *knowledge management system*).

El projecte Plone va començar el 1999. El 2004 es va crear la Fundació Plone per a protegir i promoure l'ús de Plone.

Curiositat

L'estil visual *monobook* de la Viquipèdia està basat parcialment en l'estil de les pàgines de Plone.

Les **funcionalitats bàsiques** són:

- Edició *inline*.
- Control de versions.
- Revisió de la integritat dels enllaços.
- Editor HTML visual.
- Capacitats de cicle de treball.
- Capa d'autenticació.
- Indexació de documents Word i PDF.
- Cercador basat en el protocol *sitemap*.
- Suport per a *wikis*.
- Navegació endavant/endarrere automàtica.

- Generació automàtica de taules per al contingut.
- Suport per a continguts multilingües.
- Publicació diferida.
- URL nets.
- Editor gràfic.
- Compressió de recursos.
- Encauament amb integració de *proxy*.
- Reordenació de continguts amb arrossegar i deixar anar.
- Exportació de la configuració del lloc en XML.
- Format automàtic per a impressió.
- XHTML i CSS estàndard.
- Compleix criteris d'accessibilitat.
- Exportació automàtica de *feeds* RSS.
- Escalat automàtic d'imatges i generació de miniatures (*thumbnails*).
- Multiplataforma.
- Suport de microformats.
- Possibilitat d'afegir comentaris en qualsevol contingut.
- Suport per a FTP i WebDAV.
- Edició *in-context*.
- Suport per a *backups*.
- Operació de tallar/copiar/enganxar (*cut/copy/paste*) en els continguts.

Com veiem, la llista de funcionalitats és molt extensa.

Els **avantatges** principals són:

- Basat en ZODB (base de dades orientada a objectes no relacional).
- Python sol ser més fàcil de programar que altres llenguatges, com el PHP.
- Usa un bon sistema de plantilles anomenat TAL *(template attribute language*)
- És flexible.

I els **desavantatges**:

- La documentació no està gaire actualitzada.

- La importació de dades no està gaire ben resolta.

- Les tecnologies sobre les quals se sustenta encara no són gaire conegudes per la comunitat de desenvolupadors.

Exemples de llocs web que usen aquest CMS:

Lufthansa (http://www.lufthansa.com/).

Oxfam America (http://www.oxfamamerica.org/).

7.1.4. El Drupal

El **Drupal** (http://www.drupal.org) és un CMS de codi obert amb llicència GPL basat en PHP/MySQL. La comunitat que hi ha al darrere de Drupal és molt àmplia, la qual cosa permet que hi hagi molta documentació, suport per a possibles problemes i un gran nombre de mòduls addicionals.

Les **funcionalitats bàsiques** són les que donen els mòduls del nucli (*core*), els que s'instal·len automàticament sense necessitat d'afegir mòduls:

- Accés a estadístiques.
- Cerca avançada en el lloc.
- Publicar en mode blog, "Llibre", fòrums i enquestes.
- Gestió de blocs i regions de contingut.
- Encauament i *throttling*.
- URL nets.
- Sistema de menús multinivell.
- Suport per a múltiples llocs amb la mateixa instal·lació.
- Suport per a múltiples usuaris.
- Integració amb OpenID.
- Publicació en RSS.
- Possibilitat d'agregar *feeds*.
- Notificacions d'actualitzacions de seguretat.
- Perfils d'usuari.
- Rols d'usuari.
- Eines de cicle de treball.
- Multiidioma.

Els **avantatges** principals són:

- Plataforma altament verificada.
- Una mateixa instal·lació per a diversos llocs.
- S'adapta bé a aplicacions no solament de gestió de continguts, sinó també de comunitats.
- Sistema potent de plantilles. Qualsevol XHTML o CSS plantilla es pot convertir fàcilment a Drupal.
- Fàcil d'usar amb les opcions bàsiques.
- API.

I els **desavantatges**:

- Si no compleix perfectament amb els requisits del nostre projecte, l'adaptació pot ser difícil (probablement requerirà la programació de mòduls a mida).

- El panell d'administració no és gaire intuïtiu.

- És necessari buscar i provar en la llista de milers de mòduls disponibles.

- Cal anar amb compte amb el gran nombre de mòduls disponibles, ja que molts no són estables.

Exemples de pàgines que usen el Drupal:

The White House (http://www.whitehouse.gov/).

Ubuntu (http://www.ubuntu.com/).

7.1.5. Google Sites

Google Sites (http://sites.google.com/) és un cas especial, ja que, a diferència de la resta de CMS dels quals parlem, és un **servei en línia** i **no és de codi obert**. Pot ser una bona solució per a llocs no gaire complexos.

Google Sites està inclòs en el paquet Google apps que inclou solucions per a les empreses com Gmail, Calendar, Docs o Gtalk.

Google Sites permet crear un lloc web mitjançant un editor senzill i fins i tot utilitzar un domini propi i la integració amb altres serveis oferts per Google. Google Sites es pot considerar un CMS, ja que compleix dos dels requisits principals que han de complir aquest tipus d'aplicacions:

- Qualsevol persona pot crear molt fàcilment una pàgina web amb continguts actualitzables.

- Es poden usar temes intercanviables (presentació).

En un sol pas es pot crear un lloc a http://sites.google.com/.

Entre les opcions predeterminades (plantilles) tenim, per exemple, el lloc dels alumnes d'una classe, un *wiki* per a un projecte, o una pàgina familiar.

En no tenir accés al codi font del programa, no el podem ni modificar ni integrar amb altres sistemes. És una solució limitada que, no obstant això, per la seva senzillesa pot ser útil en casos puntuals. El seu funcionament està basat en la creació de pàgines i la seva edició posterior.

Les **funcionalitats bàsiques** són:

- Exportació de *feeds* RSS.
- Subscripció per correu electrònic als canvis de la pàgina.
- Integrat amb tots els serveis de Google, incloent AdSense, Google Docs, Calendar, Maps, Picasa, Analytics.
- Control de revisions dels continguts.
- Diferents *layouts* possibles.
- Possibilitat d'afegir ginys (*gadgets*).
- No es mostren anuncis (a diferència d'altres serveis de Google).

Els **avantatges** principals són:

- És una solució molt senzilla; el panell d'administració, per exemple, no és gens complex.
- Podem triar entre un gran nombre de plantilles que, en molts casos, estan implementades per altres persones.
- Si s'implementa un projecte usant Google Sites es pot "alliberar" el projecte perquè altres persones el puguin reutilitzar, passant des d'aquest moment a ser una altra plantilla de les disponibles. Aquest component social fa possible que hi hagi una gran quantitat de plantilles disponibles.

I els **desavantatges**:

- Falta de privacitat i, sobretot, les dades estan emmagatzemades en els servidors de Google i no en la nostra organització.
- Està limitat a 100 Mb d'emmagatzematge en la versió gratuïta.
- No es pot afegir CSS en els temes i l'HTML està limitat.
- No es poden afegir comentaris anònims.

Exemple:

T and M Photography (http://www.tandmphoto.co.uk/).

7.2. Fòrums

Un **fòrum** és un lloc d'Internet en què la gent intercanvia missatges en forma de discussió, fent preguntes i responent-les.

Aquestes converses se solen agrupar per temes. És una de les formes de creació dinàmica de continguts que va sorgir de manera més primerenca en la Xarxa.

Els fòrums són els descendents de sistemes més antics com els BBS (*bulletin board system*) de la dècada dels vuitanta i dels noranta i, en certa manera, són els precursors de les xarxes socials actuals.

Les persones que participen en un fòrum d'Internet es divideixen en grups d'interès a partir de debats. Cada usuari pot deixar la seva aportació com a resposta a un missatge d'un altre usuari o com una nova aportació, però mai editant les entrades d'altres usuaris (tal com passa en un *wiki*).

Un fòrum està controlat per un o diversos moderadors (*mods*), que són els encarregats que els usuaris segueixin les regles del lloc. Les aportacions dels usuaris s'agrupen en temes (*topics* o *threads*).

7.2.1. phpBB

phpBB (*PHP bulletin board*) (http://www.phpbb.com/) és un dels CMS per a fòrums més usats. La primera versió va sorgir l'any 2000 i des de llavors s'ha anat actualitzant mantenint el seu propòsit principal, la gestió d'un fòrum complet.

phpBB és un sistema de fòrums gratuït basat en un conjunt de paquets de codi programats en el popular llenguatge de programació web PHP i publicat sota la llicència pública general de GNU. La seva intenció és proporcionar fàcilment, i amb moltes possibilitats de personalització, una eina per a crear comunitats.

Les **funcionalitats bàsiques** són:

- Gratuït i de codi obert.
- Suport per a diversos servidors de bases de dades (MySQL, SQL Server, Oracle, etc.).
- Creació il·limitada de fòrums i subfòrums.
- Millora del rendiment respecte de la versió anterior.

- Ús de *caché* per a tots els arxius del fòrum.
- Registre d'usuaris i personalització de cada camp.
- Missatges privats a múltiples usuaris i carpetes de missatges.
- Cerca de temes i usuaris.
- Panell d'administrador i de moderador per separat.
- Creació d'enquestes amb múltiples opcions.
- Perfil per a cada usuari amb les seves dades personals.
- Aplicar "Ban" per temps definit o indefinit.
- Els usuaris poden tenir "Amics" o "Ignorats". Els missatges d'"Ignorats" s'oculten automàticament.
- Personalització de BBCode.
- Creació de grups d'usuaris, moderadors o administradors.
- Advertiment i reports d'usuaris a moderadors davant d'apunts (*posts*) indeguts.
- Possibilitat de crear nous camps per al perfil d'usuari.
- Possibilitat d'editar, des del panell d'administració, els arxius del tema usat.
- Múltiples arxius adjunts.
- Modificacions i estils gratuïts desenvolupats per la comunitat.
- Fàcil creació i assignació de rangs per apunts o per grups.
- *Log* d'accions d'usuaris, moderadors i administradors.

Els **avantatges** principals són:

- És una eina molt completa per al seu propòsit, difícilment trobarem un CMS que tingui més funcionalitats en aquest sentit.
- La comunitat aporta MOD, modificacions que estenen les funcionalitats o que canvien la presentació.

I els **desavantatges**:

- Si el nostre projecte es desvia del propòsit principal d'un fòrum, no serà la solució òptima.
- Hi ha multitud de CMS orientats a la gestió de fòrums a Internet, phpBB només n'és un.

Pàgina web

Comparativa de CMS per a fòrums basats en PHP:
http://en.wikipedia.org/wiki/Comparison_of_Internet_forum_software_ %28PHP%29.

7.3. Blogs

Un **blog** és una publicació electrònica que recopila articles publicats per un o més autors ordenats cronològicament de més a menys recent.

Normalment, els visitants del blog poden deixar comentaris. La majoria de blogs publiquen automàticament els continguts amb el format RSS o Atom, de manera que la gent (entre altres coses) es pot subscriure a la publicació.

El gran creixement de la blogosfera observat els últims anys (des de mitjan anys noranta) ha estat propiciat principalment per l'eclosió de diversos CMS que han facilitat enormement la gestió d'aquests blogs. Actualment hi ha milions de blogs de diversos tipus, tant personals com corporatius, que s'usen com a eina de comunicació entre les empreses i els clients.

Altres funcionalitats que ha d'incloure qualsevol blog són:

- Sindicació en RSS/Atom de les entrades i els comentaris.
- *Ping*. Mecanisme pel qual el blog avisa determinats servidors que s'ha publicat un nou contingut.
- *Blogroll*. Possibilitat de mantenir una llista d'enllaços a altres blogs que l'autor considera interessants.

Els ***podcasts*** i els ***vodcasts*** són variants de blogs en què el mitjà principal no és el text, sinó l'àudio (*podcast*) o el vídeo (*vodcasts*). A efectes pràctics, funcionen exactament igual que un blog de text. Els usuaris es poden subscriure als continguts usant agregadors, encara que en aquest cas poden ser agregadors específics de *podcasts*, com per exemple l'iTunes d'Apple, o reproductors de vídeo a Internet com Miro.

7.3.1. El WordPress

El **Wordpress** (http://www.wordpress.org/) és una de les plataformes CMS més utilitzades en blogs i també en projectes web més generals. Actualment es troba en la versió 3. Des del principi es va plantejar com un sistema perquè els autors de blogs en poguessin administrar el contingut d'una manera senzilla, però ha evolucionat tant que actualment aquest CMS s'usa en llocs web de propòsit més genèric.

Les **funcionalitats bàsiques** són:

- Sistema de ginys (*widgets*) que permet afegir funcionalitats mitjançant arrossegar i deixar anar.
- Activació senzilla de temes (*themes*).
- *Plug-ins* que afegeixen funcionalitats.
- Instal·lació i actualització de temes i *plug-ins* directament des del panell d'administració (sense haver de descarregar arxius).
- Múltiples categories per als articles.
- Definició de diferents tipus de contingut.
- *Trackbacks* i *pingbacks*.
- Aplicacions natives per a actualitzar el WordPress des d'Android i iPhone.
- Actualització remota del blog.

Els **avantatges** principals són:

- Enfocat a un ús senzill. El panell d'administrador és molt intuïtiu.
- Optimitzat per a cercadors (SEO).
- Les pàgines de WordPress poden tenir "pares", de manera que es pot crear una jerarquia (seccions i subseccions).
- Hi ha infinitat de *plug-ins* que afegeixen funcionalitats.

I els **desavantatges**:

- Encara que és bastant flexible, si les necessitats del lloc van més enllà de les pròpies de WordPress pot ser problemàtic; no és, per exemple, la millor opció per a un lloc de comerç electrònic.

- No és la millor opció per a un lloc amb milers de pàgines (pel seu rendiment).

Exemple:

Ràdio Associació de Catalunya. RAC1 (http://www.rac1.org/).

7.3.2. PivotX

PivotX (http://pivotx.net/) és un altre exemple de CMS orientat a la creació de blogs, també amb llicència GPL. Una de les particularitats que té és que **no necessita un motor de base de dades** (pot emmagatzemar la informació en un arxiu), factor que facilita la instal·lació en entorns en què no tinguem disponible cap sistema de base de dades.

El sistema de plantilles (temes) que usa està basat en Smarty, sistema de plantilles web basat en PHP. A més, usa altres projectes com jQuery i tinyMCE (editor HTML).

Les **funcionalitats bàsiques** són:

- Sistema de plantilles basat en Smarty.

- Múltiples blogs amb una sola instal·lació.

- Múltiples autors.

- Possibilitat d'incrustar imatges i fins i tot usar l'extensió picnik, que permet editar les imatges.

- Extensions.

- Possibilitat d'usar un arxiu per a emmagatzemar les dades o una base de dades MySQL.

- Sistema antiinundació (*spam*).

- Optimitzat per a cercadors (SEO).

- Basat en un *framework* que permet crear extensions.

Els **avantatges** principals són:

- És senzill.

- És una bona opció si no disposem de base de dades.

I els **desavantatges**:

- La comunitat de PivotX encara és molt incipient.
- Poques extensions i temes disponibles.

Observació

Aquest CMS encara no està prou estès per a proporcionar bons exemples.

7.4. *Wikis*

Un *wiki*[1] és un lloc web en què els visitants poden editar directament i fàcilment qualsevol de les seves pàgines i afegir seccions noves.

(1)Del hawaià *wiki*: 'fer les coses de manera senzilla i ràpida'.

Molts *wikis* permeten editar el contingut sense estar ni tan sols registrat. En qualsevol cas, es guarda l'historial de tots els canvis i l'autor (o, si és anònim, el seu IP), per a poder recuperar en qualsevol moment un estat anterior de qualsevol pàgina.

El *wiki* més conegut és la **Viquipèdia**, que està implementat en el CMS *wiki* de codi obert MediaWiki.

En la majoria de *wikis* el text s'edita en text net (en lloc d'utilitzar HTML) per a fer-lo més fàcil d'utilitzar. Algunes convencions especials del text serveixen per a afegir caràcters especials al text.

Per exemple, és el cas del CamelCase, una convenció que permet definir enllaços entre pàgines escrivint diverses paraules sense espais i marcant amb majúscules la primera lletra de cada paraula:

AixòÉsUnExemple

D'aquesta manera, l'aplicació de *wikis* sap que és un enllaç a una altra pàgina i ho tradueix com un enllaç d'HTML.

7.4.1. MediaWiki

MediaWiki (http://www.mediawiki.org/wiki/MediaWiki) funciona sota la plataforma PHP i està llicenciat en GPL.

Es pot escriure en MediaWiki en un format especial anomenat *wikitext*, que permet editar fàcilment i donar format al text sense instruccions d'HTML o CSS. El desenvolupament de MediaWiki està patrocinat per la fundació Wiki-Media.

Les **funcionalitats bàsiques** són:

- Es poden definir tipus de pàgines.
- Cada pàgina del *wiki* té una pàgina de discussió pròpia.

- Suport de TeX per a visualitzar fórmules matemàtiques.
- Sistema de *plug-ins* que permet estendre fàcilment les funcionalitats.
- Capacitat de bloquejar temporalment usuaris o pàgines.
- Suport de plantilles personalitzades amb paràmetres.
- Creació de línies de temps mitjançant codi *wiki*.
- Sistema de categories jeràrquic, que permet crear llistes d'articles o de miniatures d'imatges.
- Admet diversos nivells d'usuari.
- Suport per a emmagatzemament de memòria virtual o cau (*caché*), també coneguts com *memcached* i el sistema de *caché* Squid.
- *Skins* personalitzables per cada usuari.

Els **avantatges** principals són:

- A diferència dels *wikis* clàssics, no és necessari que els noms de les pàgines estiguin en CamelCase, la qual cosa permet tenir noms més naturals.
- És un sistema altament provat, ja que és la base de la Viquipèdia.

I els **desavantatges**:

- Un *wiki* és una bona eina de treball col·laboratiu; no obstant això, no és la millor opció per a personal amb coneixements a nivell d'usuari perquè el format de text requereix un temps d'aprenentatge.
- Cal tenir en compte que l'estructura d'un *wiki* s'adapta només a cert tipus de projectes i que requereix coneixements avançats d'edició (almenys més alts que en CMS, en què s'actualitza el text en camps sense cap tipus de format).

  Exemple:

  Wikipedia (http://es.wikipedia.org/).

7.4.2. Tiki Wiki

Tiki Wiki (http://info.tikiwiki.org/) és un altre CMS enfocat a *wikis* de codi obert, amb llicència GPL i basat en PHP. Per diferenciar-se de la resta, vol cobrir altres àrees d'ús com webs genèrics, portals, intranets o extranets i, més en particular, l'àrea del treball col·laboratiu.

Les **funcionalitats bàsiques** són:

- Possibilitat de crear diferents tipus de contingut com articles, fòrums, butlletins, blogs, galeries d'imatges i, per descomptat, *wikis*.
- Possibilitat de crear dibuixos mitjançant una miniaplicació (*applet*) de Java.
- Rastrejadors.
- Sondejos, enquestes i qüestionaris.
- FAQ.
- PMF.
- Xat.
- Sistema de gestió de bàners.
- Integració amb sistemes de correu.
- Calendari.
- Mapes i gràfics.
- Versió per a mòbils: Tiki mòbil.
- *Feeds* RSS.
- Sistema de categories.
- Activació de temes.
- Flux de treball.
- *Shoutbox*.
- ACL (sistema estàndard de permisos sobre objectes en una xarxa).

Els **avantatges** principals són:

- No solament és un CMS per a *wikis*, sinó que aporta diverses eines útils per al treball col·laboratiu; possiblement, és un dels CMS més complets en aquest sentit.

- Pot funcionar amb un gran nombre de servidors de bases de dades.

I els **desavantatges**:

- El rendiment i l'escalabilitat necessiten millores.

- *Featuritis*. Tiki Wiki ofereix tantes funcionalitats que és difícil que totes es puguin ajustar bé als nostres requisits.

- La gran quantitat de funcionalitats que implementa fa difícil la traducció de totes als diferents idiomes.

Exemple:

Mozilla Firefox Support (http://support.mozilla.com/).

7.5. Aprenentatge electrònic

En l'àmbit de l'educació i l'aprenentatge electrònic (*e-learning*) també podem trobar diversos CMS especialitzats. Les funcionalitats molt específiques que es requereixen en aquest tipus d'entorns fa realment necessari que els CMS estiguin adaptats per a cobrir aquestes necessitats. El flux de treball està molt determinat pel tipus de continguts (de tipus didàctic en forma de lliçons, unitats, proves, etc.) i els perfils (professors, alumnes).

7.5.1. Moodle

Un dels CMS per a l'aprenentatge electrònic més usats i amb una gran comunitat de suport és **Moodle** (http://www.moodle.org/), també basat en PHP/MySQL i amb llicència GPL. La comunitat de desenvolupadors de Moodle ha permès que aquest CMS estigui traduït a infinitat d'idiomes (més de vuitanta) i que s'hagin desenvolupat un gran nombre d'extensions per a cobrir necessitats específiques.

Les **funcionalitats bàsiques** són:

- Vénen donades pels mòduls.
- Tasques i qualificacions per als alumnes.
- Consultes. Funcionen com una votació.
- Fòrum. Exclusius per a professors, per a alumnes o per a tots.
- Diari. Informació privada entre el professor i l'alumne.
- Qüestionari. Definició de preguntes en qüestionaris.

- Recursos. Continguts digitals en format Word, PowerPoint, vídeo, etc.
- Enquestes.
- *Wiki*. Per a treball col·laboratiu en documents.

Els **avantatges principals** són:

- Hi ha una gran comunitat darrere de Moodle, la qual cosa fa que hi hagi una gran quantitat d'extensions i que molts problemes es puguin solucionar buscant informació a Internet.

- És molt configurable gràcies a l'existència de molts *plug-ins*.

I els **desavantatges**:

- No hi ha un sistema d'avisos general per a tots els cursos, és necessari anar d'un en un per a veure'ls.

- El control de grups d'estudiants és per curs, no general.

- La instal·lació es podria simplificar.

7.6. Xarxes socials

Les **xarxes socials** són un tipus d'aplicació cada vegada més comú a Internet; Facebook, Twitter o MySpace només en són alguns exemples. Aquest fet ha generat la necessitat de crear aplicacions d'aquest tipus a mida, el que podríem anomenar *aplicacions de xarxes socials*.

Una **xarxa social** és una aplicació web en què tots els usuaris queden identificats mitjançant perfils (dades bàsiques, fotos, etc.) i poden generar continguts i establir relacions entre ells (xarxa de contactes, missatges interns, etc.).

Encara que els CMS genèrics (com Drupal o fins i tot WordPress) es poden utilitzar com a base per a crear una xarxa social d'aquest tipus (mitjançant la instal·lació d'alguns mòduls especials), hi ha CMS que de partida ja estan pensats per a aquest propòsit. Hi ha alternatives d'aplicacions en línia com Ning o SocialEngine, encara que es tracta de solucions que són propietàries i de pagament. Alguns d'aquests CMS de codi obert són:

- **Pligg** (http://www.pligg.com/). Implementa una xarxa social basada en un sistema de recomanació (de tipus "digg.com" o "meneame.com"). És un CMS complet amb possibilitat d'afegir mòduls.

- **Meneame.com** (http://www.meneame.com). Basat en el mateix principi, també és una eina de codi obert i de fet ja hi ha clons d'aquesta aplicació en altres idiomes o amb altres temàtiques.

- **Elgg** (http://elgg.org/). És una eina més completa, ja que no sols és una xarxa de recomanacions, sinó que el seu objectiu és poder funcionar com a base d'una xarxa dins d'una organització (dins d'una universitat, per exemple, o simplement un grup de gent amb els mateixos interessos).