Introducció a les aplicacions web


1. Tipus d'aplicacions

Quan estem utilitzant un ordinador, una tauleta o un telèfon mòbil, quins tipus d'aplicacions o programes podem estar utilitzant? Bàsicament distingim dos grans grups:

- Aquelles aplicacions que no necessiten cap connexió a Internet o a una xarxa d'ordinadors per a funcionar. Aquest tipus d'aplicacions solen dir-se **aplicacions d'escriptori**, i podem trobar exemples molt variats: un processador de textos, un lector de llibres electrònics, un reproductor de música o vídeo, i fins i tot videojocs que tinguem instal·lats.
- Aquelles aplicacions que sí que necessiten connexió, bé siga a Internet o a un ordinador de la seua xarxa local. En aquest altre grup també tenim exemples variats d'aplicacions. Per exemple, si compartim un document de text en Google Drive, o si obrim el navegador per a accedir a una plataforma d'un curs en línia, o fins i tot si juguem a videojocs juntament amb altres persones d'altres llocs. Ací distingim diversos subtipus d'aplicacions. Alguns dels més habituals són:
  - Les **aplicacions P2P** (**peer-*to-*peer*), on tots els elements connectats a la xarxa tenen el mateix "rang", per dir-ho així, i comparteixen informació entre ells. És el mecanisme en el qual es basen diversos programes de descàrrega, com els d'arxius tipus **torrent*.
  - Les **aplicacions client-servidor**, dites així perquè consisteixen en el fet que un conjunt d'ordinadors (anomenats *clients*) es connecten a un central (anomenat *servidor*) que és el que els proporciona la informació i els serveis que sol·liciten. En el cas de videojocs on ens connectem a un altre lloc per a jugar amb altres persones, estem utilitzant aplicacions client-servidor, on el nostre ordinador (un dels clients), té instal·lada una part de l'aplicació i el servidor al qual es connecta li proporciona la informació dels escenaris i de la resta de jugadors i personatges.
    - Dins del tipus d'aplicacions client-servidor, les **aplicacions web** són un subtipus, potser el més nombrós. És en aquest subtipus en el qual ens centrarem.

1.1. Què és una aplicació web?

Podem trobar diverses definicions d'aplicació web buscant en Internet. Una de les més habituals fa referència a aplicacions que es carreguen o executen des d'un navegador web, accedint a un servidor.

No obstant això, l'avanç experimentat en aquest sector durant els últims anys fa que puguem definir un concepte més ampli. Així, una **aplicació web** seria aquella realitzada a partir de llenguatges i tecnologies de desenvolupament web, com ara HTML, CSS, JavaScript, PHP... Poden executar-se en un navegador convencional, o un motor de navegador embegut en un altre sistema, com el *webview* d'Android o iOS, per exemple. WebView és un component que permet a les aplicacions mostrar contingut web sense haver d'obrir un navegador. Té un ús més limitat que aquest però, en estar pensat per a consultes puntuals des de la pròpia aplicació, la càrrega de pàgines és més ràpida. Ho usen aplicacions com Facebook, Instagram, etc.

D'aquesta manera, a més de les aplicacions web "tradicionals", també tindrien cabuda les anomenades aplicacions *híbrides* (desenvolupades amb tecnologies web però exportades a format natiu de diversos dispositius), i aplicacions d'escriptori que empren tecnologies web, com per exemple el framework Electron de JavaScript.

2. Arquitectura d'una aplicació web

2.1. Què és "la web"?

Podem veure la web com una espècie de plataforma mundial on tenim disponibles gran quantitat de recursos (documents, videojocs, xarxes socials, fòrums, etc.). Es va fer popular a principis dels anys 90 gràcies a aplicacions com el correu electrònic, els xats, etc. i amb l'aparició de la web 2.0 van vindre una altra sèrie d'aplicacions que la van potenciar encara més, com els blogs o les xarxes socials. A poc a poc s'han anat afegint funcionalitats, fins al punt que fa pocs anys era impensable poder veure vídeos o pel·lícules en Internet, i hui és una cosa molt habitual.

2.2. Elements d'una aplicació web

En una aplicació web podem distingir en primer lloc dos grans costats: el **client**, on està l'usuari, que utilitza normalment un navegador web (Google Chrome, Firefox, etc.) per a accedir a l'aplicació, i el **servidor**, on està situada l'aplicació (el fòrum, la xarxa social, el blog, el curs en línia, etc.), i que s'encarrega d'atendre les peticions dels clients i proporcionals la informació que sol·liciten.

2.3. Funcionament d'una aplicació web

Com hem comentat, les aplicacions web són un tipus d'aplicacions client-servidor. Aquest tipus d'arquitectures distribueixen les tasques entre els qui presten els recursos i serveis (els servidors) i els qui els sol·liciten (els clients).

Els passos que se segueixen en la comunicació client-servidor, són, normalment:

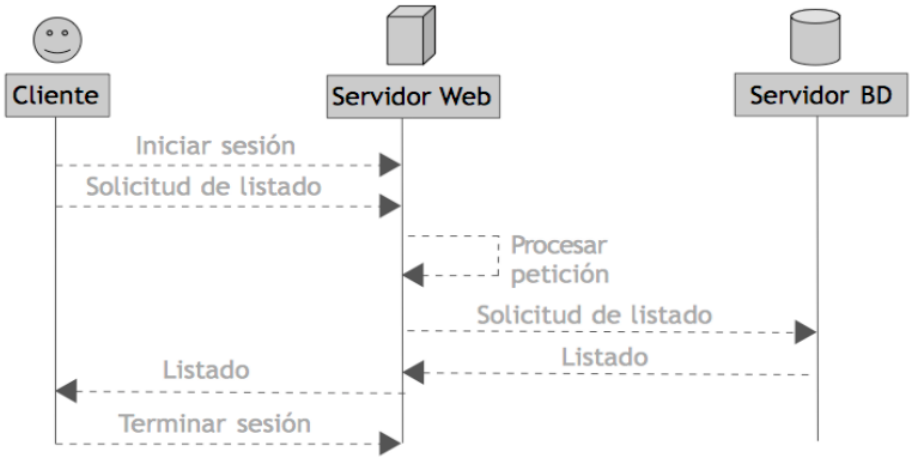
1. El client inicia sessió en el servidor
2. El client sol·licita al servidor el recurs o servei que vol utilitzar (una pàgina web, un document, pujar informació, etc.)
3. El servidor rep la petició del client, la processa i decideix quin programa ha de donar-li servei, enviant la petició a aquest programa.
4. El programa responsable processa la petició, prepara la resposta i el lliurament al servidor.
5. El servidor envia la resposta al client
6. El client pot tornar al pas 2 i realitzar una nova petició, o bé
7. El client acaba la sessió en el servidor.

En general, el servidor no té per què executar-se només, sinó que podem tindre diferents aplicacions en diferents equips (o en el mateix), la qual cosa es coneix com a **arquitectura multicapa o multinivell**. Per exemple, un servidor de bases de dades en una màquina, un servidor web en una altra (o en la mateixa que el

de bases de dades), un servidor de correu electrònic... i així distribuir els processos i el treball a realitzar, i fins i tot configurar opcions de seguretat i rendiment separades per a cada servidor.

Exemple: arquitectura de dos o tres nivells

Per exemple, si el client connectara amb el servidor per a demanar un llistat de notícies emmagatzemades en una base de dades, expressat com un diagrama de seqüència, el funcionament bàsic d'aquesta petició (i de l'arquitectura client-servidor en general) pot veure's com una cosa així:



En aquest exemple, el servidor web i el servidor de bases de dades podrien estar instal·lats en la mateixa màquina o en màquines separades, cadascuna amb el seu maquinari específic i control d'accés d'usuaris específic. En qualsevol cas, estem parlant d'una arquitectura de **tres nivells** (client, servidor web i servidor de base de dades), que és una cosa bastant habitual en les aplicacions web, perquè quasi totes compten amb una base de dades amb informació que consultar i modificar.

Sense el servidor de base de dades, estaríem davant una arquitectura de **dos nivells**, on el servidor és polivalent, i pot respondre directament a les peticions dels clients sense consultar amb altres servidors o aplicacions. Aquesta opció és menys flexible, menys segura i pot oferir pitjor rendiment en sistemes congestionats, al no poder dividir el treball entre diferents tipus de servidors.

2.4. Dominis y URLs

Hostings y noms de domini

El servidor és el component d'una aplicació web que s'encarrega de rebre peticions de tots els clients que es connecten a ell i enviar-los la informació que sol·liciten. Per a poder fer això, el servidor ha d'estar accessible en un lloc conegut, perquè els usuaris puguen connectar-se a ell. Per exemple, quan escrivim l'adreça *www.google.es*, d'alguna forma hi ha "alguna cosa" en Internet que sap on està el servidor per al cercador Google en espanyol, i envia la petició allí.

Per tant, en primer lloc, hem de localitzar el nostre servidor en Internet en el qual allotjar el nostre lloc web. Això pot fer-se de diverses formes. Per exemple:

- Disposar d'un servidor (o servidors) propis i una adreça IP pública fixa a la qual accedir. Aquesta opció és poc habitual hui dia

- Contractant un espai (o una màquina sencera, si el projecte és gran i es disposa de pressupost suficient) en una empresa d'allotjament o **hosting**. Aquesta opció és molt més habitual, i d'aquí ve que proliferen les empreses que es dediquen a l'hosting, com ara OVH, Hostinger, Ionos, etc. A més, hem de reservar (comprar) un **nom de domini** per a la nostra empresa o web. El nom de domini és el nom únic i exclusiu que se li dona a un lloc web en Internet. En el cas anterior, el domini seria *google.es*. El preu de mantindre aquest domini pot variar, però ronda els 10 o 15 euros a l'any.

L'adquisició d'un domini en Internet es diu **registre de domini**. Per a això, l'usuari ha de contactar amb una empresa registradora autoritzada per **ICANN** (*Internet Corporation for Assigned Names and Numbers*) al seu país i es comprova en primer lloc que el domini desitjat no pertany a ningú. Una vegada acceptades les condicions, l'empresa registradora contacta amb la ICANN i realitza els tràmits. D'aquesta manera, en unes hores el domini estarà disponible. Aquestes empreses acreditades com a registradors de domini han de competir entre elles per a oferir els preus més baixos o les millors promocions.

URLs

Hem vist que, en l'esquema de funcionament d'una aplicació web, el client sol·licita recursos al servidor. La forma en què els sol·licita és mitjançant URLs. Una URL (*Uniform Resource Locator*) és una seqüència de caràcters que segueix un estàndard i que permet denominar recursos dins d'Internet perquè puguen ser localitzats. Per exemple, quan escrivim en un navegador una adreça com *http://www.miweb.com/paginas/pagina.html*, estem introduint una URL per a localitzar un recurs (en aquest cas, una pàgina HTML).

Una URL es compon de:

protocol://subdomini.domini.com/carpeta/pagina?param=valor

- El **protocol**, que indica les regles que se seguiran per a comunicar-se client i servidor. Veurem més endavant alguns exemples de protocols, però per al que ens importa, en una URL el protocol va al principi, fins als dos punts i el delimitador //. En el nostre exemple, el protocol seria **http** (*HiperText Transfer Protocol*)
- El **nom de domini**, que ja hem explicat anteriorment. Identifica al servidor i l'empresa/web a la qual connectarem. Va just darrere del protocol, fins a la següent barra. Normalment acaba en *.com, .és, .net*, etc. En el nostre exemple seria *www.miweb.com*
- El **subdomini**, que és opcional, és una manera de tindre un lloc web relacionat, com a annex, a una web principal. És habitual que les empreses d'hosting permeten registrar subdominis, encara que algunes ho oferisquen amb unes certes restriccions, ja siga pel nombre de subdominis permesos o pel servei que presten. Per exemple, si tenim el domini *www.miweb.com* de l'exemple anterior com a URL principal, podríem tindre a més, *tienda.miweb.com* com a subdomini per a una botiga associada a la web.
- La **ruta cap al recurs**, que comprén totes les carpetes i subcarpetas (si n'hi ha) i el nom d'arxiu que volem obtindre. En el nostre exemple, la ruta seria */pàgines/pàgina.HTML*
- Uns **paràmetres** opcionals, que serveixen als programadors per a incloure informació addicional que utilitzen les pàgines web per a enviar informació als servidors, de manera que és possible donar ordres o afegir informació d'un formulari.

> **Exercici 1:**

> Com hem vist, els agents registradors de domini s'encarreguen de donar d'alta els noms de domini en Internet amb diferents plans de pagament per al servei. Cerca en Internet una llista d'agents registradors de domini oficial per a Espanya i consulta els preus que tenen alguns d'ells.

3. Protocols més utilitzats

A l'hora de comunicar clients i servidors, és necessari establir un **protocol** de comunicació, és a dir, una sèrie de regles que indiquen quin tipus de missatges s'intercanviaran, en quina ordre i quin contingut tindrà cada tipus de missatge, de manera que els dos extrems de la comunicació (client i servidor) puguen entendre's.

Totes les comunicacions en una xarxa (o en Internet) es basen en el protocol **TCP/IP** per a funcionar. Aquest protocol està basat en dues parts: el protocol TCP (que estableix com ha d'estructurar-se i fraccionar-se la informació per a ser enviada) i el protocol IP (que estableix com s'identifiquen els equips en la xarxa, mitjançant adreces IP).

Sobre aqueixa base de TCP/IP, s'estableixen una sèrie de protocols que s'empren segons el tipus d'aplicació que es vaja a utilitzar (web, correu electrònic, pujada d'arxius, etc). En treballar amb aplicacions web, els protocols de comunicació més emprats són:

- **HTTP** (*HyperText Transfer Protocol*), un protocol existent des de 1990 i que permet la transferència d'arxius en general, encara que principalment d'arxius HTML (és a dir, documents web). Se segueix un esquema de peticions i respostes entre client i servidor com el vist anteriorment.
- **HTTPS**, versió segura del protocol anterior, on les dades de les peticions i les respostes s'envien encriptats, perquè ningú que intercepte la comunicació puga descrifrar el contingut d'aquesta. Aquest tipus de protocols se sol utilitzar en sistemes bancaris, plataformes de pagament (Paypal, per exemple), i altres aplicacions que manegen informació delicada (DNIs, nombres de targetes de crèdit, etc.).

Normalment, els navegadors web canvien automàticament del protocol HTTP a HTTPS en connectar amb pàgines que necessiten ser més segures (login, dades de pagament, etc.). Es pot comprovar el canvi mirant la barra de direcció del navegador: en accedir al protocol segur es mostrarà el protocol *https* en la barra, o bé la icona d'un cadenat. No obstant això, sí que haurem de configurar el nostre servidor web per a acceptar comunicacions HTTPS, si fora el cas.

Altres protocols són menys utilitzats a l'hora de treballar amb aplicacions web, però sí que s'utilitzen igualment en altres aplicacions que requerisquen d'Internet. Per exemple, per a l'enviament i recepció de correu electrònic s'empren els protocols **SMTP** o **POP3/IMAP**, respectivament. Per a enviar arxius a un servidor remot es pot emprar (a més del propi protocol HTTP) el protocol **FTP**. Aquest últim protocol també és habitual a l'hora de treballar amb aplicacions web, especialment quan les estem desenvolupant, per a pujar les actualitzacions de l'aplicació al servidor.

3.1. Més sobre HTTP/HTTPS

El protocol sobre el qual es basen les aplicacions web per a funcionar és, per tant, HTTP (o la seua versió segura, HTTPS). En tots dos casos, es tracta d'un protocol per a aplicacions client-servidor, on les dades que s'envien l'un i l'altre tenen un format determinat.

Peticions HTTP

D'una banda, estan les dades que el client envia al servidor, i que es denominen **peticions** (en anglés, *requests*). Aquestes peticions es componen, a grans trets, de:

- La **URL** del recurs sol·licitat
- Unes **capçaleres de petició** que donen informació sobre el recurs sol·licitat i el client que el sol·licita. Per exemple, podem obtindre el navegador que s'està utilitzant en el client, l'idioma, etc.
- Unes **dades** addicionals, en cas que siguen necessaris. Per exemple, en el cas d'enviar un formulari o pujar un fitxer al servidor, aquestes dades poden consistir en la informació introduïda en el formulari, o en els propis bytes del fitxer a pujar, respectivament.

Totes aquestes dades, a baix nivell, s'encapsulen en paquets i es fragmenten d'acord amb el protocol TCP per a ser enviats al servidor.

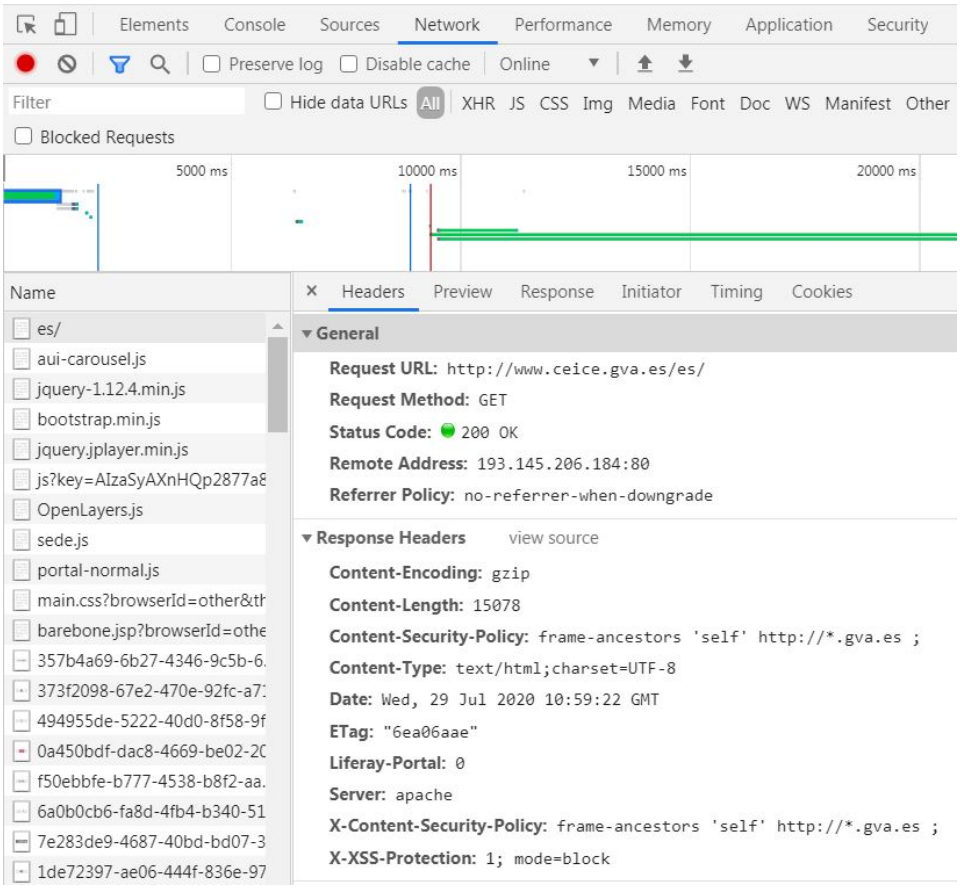
Respostes HTTP

Per part seua, el servidor, quan rep una petició d'un client, emet una **resposta** (en anglés, *response*) amb la informació sol·licitada, o amb algun codi d'error en cas que haguera succeït algun. Les respostes es componen d'aquests elements:

- Un **codi d'estat**, que indica si s'ha pogut atendre correctament la petició o no. Aquests codis estan agrupats en categories, de manera que, per exemple:
- Els codis 2xx indiquen una resposta satisfactòria. El normal és rebre un codi 200 si tot ha anat bé
- Els codis 3xx indiquen que hi ha hagut algun tipus de redirecció: el recurs sol·licitat estava en una altra URL i se'ns ha redirigit a ella
- Els codis 4xx indiquen un error per part del client. Per exemple, l'error 404 és molt típic, i indica que la URL indicada no existeix. L'error 403 és també típic, i indica que el client no té permís per a accedir al recurs sol·licitat.
- Els codis 5xx indiquen un error per part del servidor. Per exemple, que estiga col·lapsat i excedisca el temps d'espera per a atendre la petició.
- Unes **capçaleres de resposta**, que donen informació sobre la resposta que s'envia. Per exemple, la grandària de la resposta, el tipus de contingut (si és un document web, un arxiu ZIP, etc... és el que es coneix com a *tipus ACARONE*), l'última data de modificació, etc.
- El **contingut** sol·licitat, si no hi ha hagut error en tramitar la petició. Per exemple, el contingut de la web que s'ha sol·licitat, o d'un arxiu que s'ha demanat descarregar.

Monitoratge amb Google Chrome

Si tenim a mà el navegador Google Chrome, podem comprovar el que client (navegador) i servidor s'envien en un procés HTTP. Per a això, anem al menú d'Eines per a desenvolupadors, i més concretament a la secció Network. Des d'ací, accedim a una web coneguda (per exemple, *ceice.gva.es*), i podem comprovar la informació enviada i rebuda:



> **Exercici 2:**
>
> Utilitza Google Chrome (opció de *Eines per a desenvolupadors*, pestanya *Network*) per a veure l'esquema de petició i resposta HTTP cap a alguna web coneguda, com per exemple *stackoverflow.com*. Identifica el codi d'estat, les capçaleres de resposta (tracta d'identificar algunes d'elles) i el contingut.

4. Sistema de noms de domini o DNS

En les comunicacions en xarxes TCP/IP, quan un equip envia un paquet de dades a un altre, és necessari identificar tant l'origen com la destinació. Les aplicacions que inicien la transmissió han d'incorporar valors obligatoris com ara: adreça MAC d'origen i de destinació, adreça IP d'origen i de destinació i port d'origen i de destinació. Aquests camps sempre han de tindre un valor assignat. Això significa que, quan es vol visitar una pàgina web que està allotjada en un equip (per exemple, l'equip amb el nom *www.google.es*), el client ha de saber quina és l'adreça IP de destinació per a incorporar-la en el paquet.

De fet, quan s'obri un navegador web és possible accedir a un recurs tant pel seu nom com per la seua adreça IP. Seguint amb l'exemple anterior, si accedim a una finestra de terminal i executem la comanda *ping www.google.es*, obtindrem en la primera línia la seua adreça IP. Si es col·loca aqueixa mateixa adreça IP en un navegador web, es pot comprovar que retorna la mateixa pàgina web d'inici. No obstant això, com a usuaris, ens resulta més fàcil dirigir-nos a un equip (host, servidor web, servidor de correu, etc.) utilitzant el seu nom que recordar la seua adreça IP corresponent.

El **DNS** (*Domain Name System*) o **Sistema de Noms de Domini** és l'encarregat de proporcionar un mecanisme eficaç per a la traducció d'adreces IP de recursos de xarxa a noms fàcilment llegibles i memorizables per les persones, i viceversa. A aquesta acció se la coneix com a *resolució DNS*. Per a això, utilitza una base de dades distribuïda i jeràrquica que associa adreces IP d'hosts, serveis o qualsevol recurs connectat a internet o xarxa privada amb informació de divers tipus.

- **Jeràrquica** perquè s'organitza en una estructura de dominis, els quals es componen de subdominis, que al seu torn es poden dividir en altres subdominis i així fins a 127 nivells.
- **Distribuïda** perquè la informació de la base de dades no està tota en un solo servidor central, sinó que la informació es troba repartida per parts en diferents servidors DNS d'Internet.

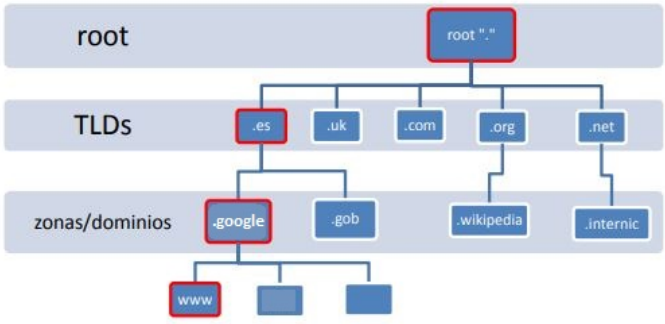
Suporta tant IPv4 com IPv6, i la informació s'emmagatzema en forma de registres de recursos, en anglés *Resource Rècords* (RR), de diferents tipus els quals poden emmagatzemar adreces IP o un altre tipus d'informació. Aquesta informació s'agrupa en zones, que corresponen a un espai de noms o domini concrets i que són mantingudes pel servidor DNS autoritativo d'aquesta. A continuació, aprofundirem una mica més en aquests i altres conceptes.

4.1. Elements integrants del DNS

El DNS s'estructura en tres components principals: *espai de noms de domini*, *servidors de noms* i *resolvers*.

Espai de noms de domini

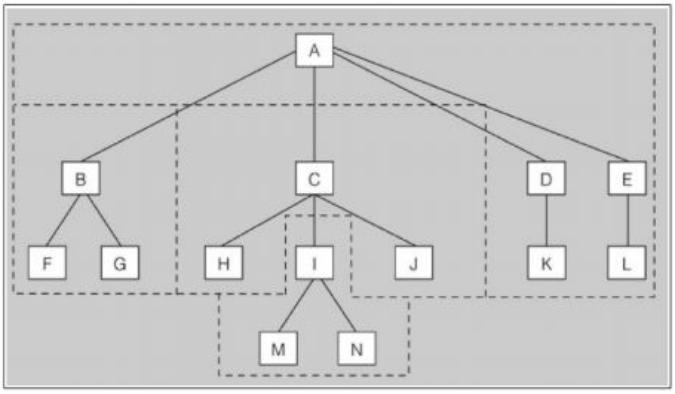
El **espai de noms de domini** consisteix en una estructura jeràrquica d'arbre invertit on cada node conté **registres de recursos** (*Resource Rècord, RR*) que proporcionen informació sobre el component maquinari i programari que recolzen aquest domini, com per exemple, els hosts, els servidors de noms, els servidors web, els servidors de correu, etc. Del node arrel, situat en el nivell més alt, parteixen les branques que conformen les zones. Aquestes, al seu torn, poden contindre un o més nodes o dominis que al seu torn poden dividir-se en subdominis segons es baixa en la jerarquia.



- Cada node de l'arbre es diu domini i rep una etiqueta o nom. El nivell més alt de tota la jerarquia és el domini arrel o root, i es representa per "." (punt). El nom de domini d'un node es crea mitjançant la concatenació de totes les etiquetes, començant per aquest node i acabant en el node arrel. Per exemple, *www.google.es.*. En el sistema DNS un node pot tindre un nom de fins a 63 caràcters. La profunditat de nodes està limitada a 127 nivells.

- Just un nivell per davall es troben els *Top Level Domains* o TLDs, els quals estan allotjats en el que es coneix com a **servidors arrel**. Existeixen diversos servidors arrel situats en diferents continents. S'identifiquen amb les lletres de l'alfabet, i diversos d'ells es troben dividits físicament i dispersos geogràficament amb la finalitat d'augmentar el seu rendiment. Els seus noms són de la forma *letra.root-servers.org*. Són gestionats per organitzacions independents. Per exemple, *a.root-servers.org* és gestionat per *VeriSign, Inc*. Pots trobar més informació sobre aquests servidors en *https://root-servers.org/*.
- Respecte als TLDs, existeixen 3 tipus:
- **Dominis de nivell superior geogràfics (*ccTLD*)**: són aquells TLDs que representen a països. Utilitzen els codis de país de dues lletres definits en estàndard ISO-31663. Per exemple, *.és* per a Espanya, *.fr* per a França, *.pt* per a Portugal, etc.
- **Dominis de nivell superior genèrics (*gTLD*)**: aquells TLD que estan oberts a qualsevol persona o organització. Estan formats per tres o més caràcters. Per exemple, *.biz* per a negocis, *.com* per a propòsits comercials, *.info* per a informació general, etc.
- **Dominis de nivell superior d'infraestructura (*uTLD*)**: aquesta categoria té un únic TLD, *.arpa*, que s'utilitza exclusivament per a la gestió de la infraestructura.
- Els TLDs, al seu torn, són nodes pare d'altres nivells inferiors que es coneixen com TLDs de segon nivell. Successivament, la jerarquia continua fins a arribar a un node final que representa un recurs. El nom format per tota la cadena es coneix com *Fully Qualified Domain Name* (*FQDN*). Per tant, tot domini complet acaba en un punt final "." que indica el final de l'espai en la zona arrel. Per exemple "*www.miweb.com*" realment és "*www.miweb.com.*", on el punt final més a la dreta representa la zona arrel, encara que aquest últim normalment no es mostra.

És important no confondre els conceptes de **zona** i **domini**. Un domini es divideix en subdominis per a facilitar la seua administració, i cada part administrada per un o més servidors DNS és una zona. El domini és l'arbre de l'espai de noms i la zona és la part de l'arbre administrada per un servidor de noms de domini concret.



En la figura anterior hi ha tants dominis com requadres de lletres agrupats en 4 zones delimitades per les línies discontínues. El nom de domini corresponent a cada zona (es nomenen segons el seu node superior) és A, B.A, C.A, I.C.A. Cadascuna d'aquestes quatre zones tindrà un o més servidors DNS per a gestionar-la.

Servidors de noms

Els **servidors de noms** són servidors encarregats de mantindre i proporcionar informació de l'espai de noms o dominis.

D'una banda, existeixen servidors que emmagatzemen informació completa per a una o diverses zones i de les quals són responsables. Es diu que són **servidors autoritativos** d'aqueixes zones en qüestió, i per tant, proporcionen respostes dels dominis per als quals han sigut configurats. L'estàndard que defineix el DNS estableix que tota zona ha de tindre, almenys, dos servidors autoritativos: el **primari**, que guarda i administra les versions definitives dels registres de recursos de la zona, i el **secundari** que guarda una còpia que és actualitzada cada vegada que es produeix un canvi. Aquesta actualització es coneix com a **transferència de zona**. El motiu és proporcionar un mecanisme de redundància, robustesa, rendiment i còpia de seguretat, ja que si el servidor de noms falla i és únic, la xarxa quedarà inoperativa.

D'altra banda, hi ha un altre tipus de servidor que emmagatzema conjunts de registres de diferents zones que obté consultant als corresponents servidors autoritativos de les mateixes (cerques recursives). Aquesta informació l'emmagatzemen localment de manera temporal (caixet) i la renoven periòdicament. Són els anomenats **servidors caixet**, i serveixen per a descongestionar servidors que reben grans quantitats de peticions. Amb aquesta organització de servidors de noms, i la seua intercomunicació, s'aconsegueix la distribució i redundància de l'espai de dominis.

Resolvers

Un **resolver** és una rutina del sistema operatiu que funciona com a intermediària entre les aplicacions, el sistema operatiu i els servidors DNS. Quan una aplicació necessita una resolució, ja siga d'un nom o d'una adreça IP, crida a aquesta rutina retornant la informació desitjada de manera compatible perquè l'host l'entenga.

4.2 Protocol DNS i procés de resolució

Protocol DNS

El protocol DNS és un protocol de la capa d'aplicació, el qual es troba definit en les RFC 1034 i RFC 1035. Usa per a les comunicacions el port 53 i, en general, usa datagrames UDP ja que requereixen menys recursos de procés i de xarxa, encara que això suposa, d'altra banda, ser més susceptible a unes certes amenaces, com pot ser el falsege d'adreces IP (*IP Spoofing*) i la suplantació de missatges consulta/resposta, entre altres.

Procés de resolució en una consulta DNS

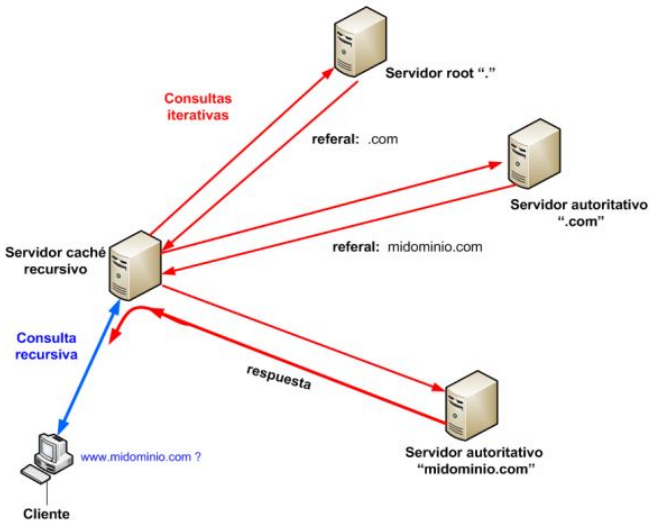
L'activitat principal d'un servidor DNS és contestar consultes, tant d'un client com d'un altre servidor DNS. Depenent de qui resolga la petició, existeixen dos tipus principals de consulta:

- **Consulta recursiva**: en la qual un *resolver* requereix al servidor DNS retornar la millor resposta basada en els seus fitxers de zona o caixet. Si el recurs sol·licitat no es troba en el propi servidor, aquest en la seua resposta retornarà un punter al servidor autoritativo del nivell més sota del domini sol·licitat, al qual ha de dirigir-se a continuació per a seguir la iteració. Per exemple, si es pregunta al servidor A, pel domini *www.midominio.com*, i el servidor A no disposa d'aqueixa informació, li contestarà amb el punter al servidor autoritativo del domini root "." perquè li sol·licite el nom. A continuació, el *resolver* continuarà la consulta iterativament, on preguntarà pel domini al servidor arrel, el qual li retornarà el punter al servidor autoritativo del domini .com. El *resolver* repeteix (itera) el procés fins que arriba al servidor autoritativo del domini desitjat on obtindrà la resposta o un error (si no existeix el registre). Normalment el *resolver*

final sol·licita consulta recursiva al servidor DNS que actua com *resolver* intermediari (caixet recursivo) evitant al client realitzar la iteració.

- **Consulta iterativa**: és aquella en què el *resolver* sol·licita del servidor DNS una resposta final o un error (si el recurs no existeix). En aquest cas el servidor DNS actua d'intermediari realitzant les consultes iteratives necessàries per a obtindre la resposta o l'error.

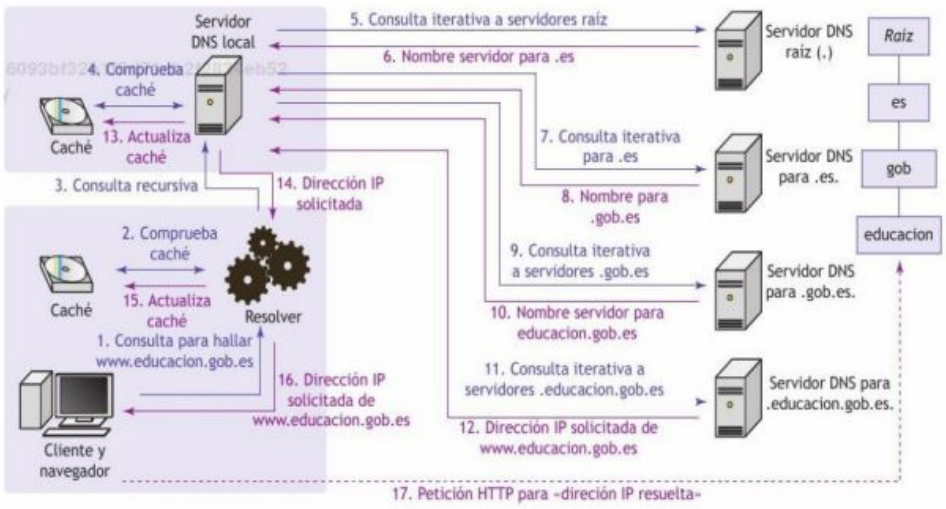
Per tant, les consultes recursives solen generar-les els clients DNS, mentre que les iteratives solen crear-les els servidors DNS quan pregunten a un altre servidor.



En termes generals, el procés que se segueix en una resolució DNS és el següent.

- El client (*resolver*) fa arribar la consulta al seu servidor DNS.
- Si el servidor DNS està configurat com autoritativo i rep una consulta DNS sobre un domini sobre la seua zona, retornarà la resposta consultant els registres emmagatzemats en la seua configuració.
- Si el servidor DNS és autoritativo i no és recursivo, i rep una consulta sobre un domini sobre el qual no és autoritativo, respondrà un missatge informant el *resolver* que no proporciona recursión i on ha de dirigir la seua consulta per a obtindre la informació autoritativa del domini sol·licitat.
- Si el servidor DNS no és autoritativo, però està configurat com recursivo i rep una consulta, aquest inicia consultes iteratives (recursión) per a trobar el servidor autoritativo del domini. Una vegada obté resposta retorna el registre al client (*resolver*) indicant que es tracta d'una resposta no autoritativa. La informació la guarda en caixet, de manera que si torna a ser preguntat pel mateix recurs, contestarà consultant la caixet.

Vegem a continuació un exemple concret de resolució de noms, per a acabar de comprendre millor tot el procés. Suposem que un client DNS necessita localitzar l'equip *www.educacion.gob.es* i per a això realitza una petició al seu servidor DNS. Suposarem també que el seu servidor no coneix aqueix domini i així veurem com es realitza la resolució completa.



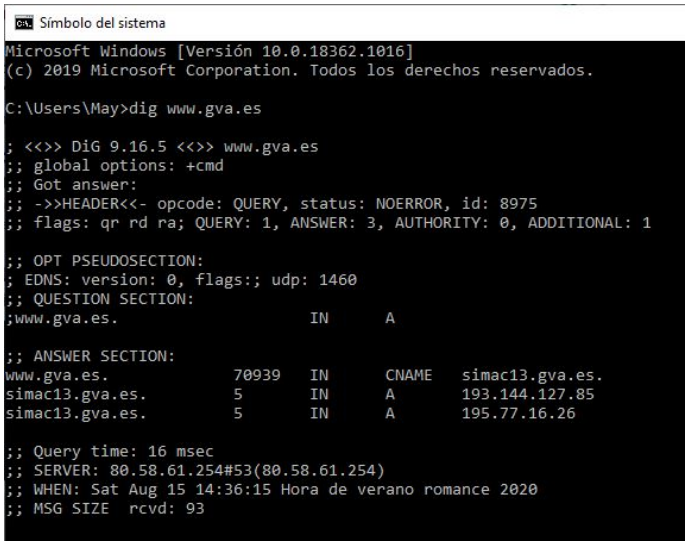
1. Un client, mitjançant un navegador web, realitza la petició de resolució de noms de l'equip *www.educacion.gob.es*. Aquesta petició es fa a través del *resolver*, que consulta el seu caixet per a veure si el té emmagatzemat.
2. Si no ho té emmagatzemat, el *resolver* inicia una *consulta recursiva* al *servidor DNS primari o local* que tinga configurat. Aquest servidor pot ser local (dins de l'empresa), o remot (en Internet). Aquest servidor mira en el seu caixet i, en cas de trobar-ho, retornarà l'adreça IP corresponent.
3. Si el servidor DNS primari no pot resoldre la consulta, aquest comença una sèrie de *consultes iteratives* per a trobar el servidor autoritativo del domini. Comença buscant en el servidor arrel. Si el servidor arrel és autoritativo per a aqueixa zona, retornarà l'adreça IP corresponent; no obstant això, si no ho és, retornarà el nom del servidor responsable per al domini de primer nivell (.és).
4. A continuació, es consulta a aqueix servidor si és autoritativo per a la zona educacion*.gob.es.* Si no ho és, retornarà el nom del servidor responsable per al domini de segon nivell (.gob.es). I així, mitjançant aquestes respostes parcials, s'arriba al servidor autoritativo, el qual retorna la IP de www.educacion.gob.es al servidor DNS local, el qual actualitza el seu caixet.
5. Una vegada actualitzada el seu caixet, el servidor DNS local envia la resposta al *resolver* del client, el qual actualitza també el seu caixet, i finalment la resposta arriba en el format adequat a l'aplicació que ha sol·licitat la resolució.

Com es pot comprovar, és un procés que consumeix bastant amplada de banda, per això cal intentar implementar una solució DNS interna que emmagatzeme en caixet totes les resolucions que es puguen en xarxes on existisca un gran trànsit cap a l'exterior, com pot ser el cas de col·legis, universitats, biblioteques o grans empreses, de manera que obtinguem un temps de resposta xicotet i, per tant, una experiència de navegació web més ràpida.

Eina DIG

DIG (*Domain Information Gopher*) és una eina de línia de comandes que realitza cerques en els registres DNS, a través dels noms de servidors, i mostra el resultat. Es pot executar tant en Linux com en Windows encara que, en aquest últim, és possible que siga necessària la seua instal·lació prèvia.

Vegem un exemple de com funciona amb la comanda `dig www.gva.es`:



Per a entendre una mica l'eixida:

- Totes les línies que inicien amb ";" són comentaris.
- La primera línia ens retorna la versió de la comanda dig que estem utilitzant. DIG 9.16.5, en el cas de l'exemple.
- Després mostra la secció de consulta amb la nostra consulta (*question section*).
- A continuació, la secció de la resposta (*answer section*). Per defecte la resposta està associada al *registre de recurs de direcció A (RR A)*, i depenent del cas, també amb el *registre de recurs de nom canònic (RR CNAME)*, però no és l'única opció. És possible consultar qualsevol altre registre de recurs (RR) del DNS. Ens centrarem en els dos registres anteriors, que són els que apareixen en la resposta de l'exemple:
- El **registre de direcció A** (*RR A*) associa noms de domini (FQDN) a direccions IPv4. El registre A té l'estructura següent: *NombreDominio IN A IP*.
- El **registre de recurs nom canònic** (*RR CNAME*) enllaça el nom de domini amb un àlies, és a dir, amb un altre nom que també redirigeix al mateix contingut. El vertader nom, per tant, és el que es connecta a través del registre A amb l'adreça IP. L'avantatge d'aquest sistema és que, si l'adreça IP canvia, només és necessari modificar el registre A. Com que tots els àlies es guien segons aquest registre A. La seua estructura és la següent: *NombreDominio IN CNAME Nom canònic o IP*
- Si ens fixem en la resposta donada, veiem que primer mostra el registre CNAME i després dos registres A. En els tres casos, els registres comencen amb el nom complet del domini, incloent-hi el punt final, seguit del temps en segons que romandrà el registre en la memòria caixet abans que la informació haja de ser sol·licitada de nou. En el cas del *registre CNAME*, es mostra a més l'àlies de *www.gva.es*, i en el cas dels *registres A* s'obté la IP de l'àlies, que en aquest cas podem observar que el domini *www.gva.es* té associades diverses adreça IP.
- Finalment, mostra estadístiques de la consulta.

Si s'executa DIG especificant l'opció **+trace** es mostra una traça pels servidors que passa fins a arribar a l'autoritari, la qual cosa permet veure la llista completa de nodes i passos de resolució d'un nom, molt útil per a entendre com funciona un sistema de noms de domini.

5. Patrons de disseny programari

Un patró de disseny o arquitectura programari comprén un conjunt de pautes a seguir, elements a desenvolupar, jerarquies i ordre que doten a una aplicació d'una estructura preestablida, que la fa més propícia per a funcionar com deu. Els seus principals objectius són, d'una banda, estandarditzar la forma en què es desenvolupen les aplicacions, i per un altre, elaborar elements o components reutilitzables entre diverses aplicacions, en ajustar-se tots a un mateix patró.

En l'àmbit de les aplicacions web, existeixen patrons de disseny específics que ens guien a l'hora d'estructurar, dissenyar i programar aquestes aplicacions. Un dels més utilitzats (o potser el més utilitzat) és el patró MVC, que comentarem a continuació, però també han sorgit uns altres (molts a partir d'aquest), que han volgut fer un volt de rosca més, o adaptar-se a les necessitats d'aplicacions web més específiques o concretes. Veurem també alguns d'aquests patrons en aquest apartat.

5.1. El patró MVC

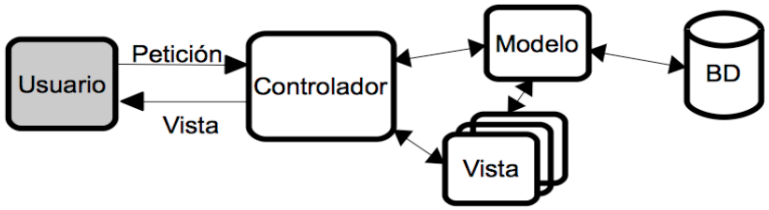
MVC són les sigles de *Model-Vista-Controlador* (o en anglés, *Model-View-Controller*), i és, com déiem abans, el patró per excel·lència ara mateix en el món de les aplicacions web, i fins i tot moltes aplicacions d'escriptori.

Com el seu nom indica, aquest patró es basa a dividir el disseny d'una aplicació web en tres components fonamentals:

- El **model**, que podríem resumir com el conjunt de totes les dades o informació que maneja l'aplicació. Típicament seran variables o objectes extrets d'una base de dades o qualsevol altre sistema d'emmagatzematge, per la qual cosa el codi del model normalment estarà format per instruccions per a connectar amb la base de dades, recuperar informació d'ella i emmagatzemar-la en algunes variables o classes determinades. Per tant, no tindrà coneixement de la resta de components del sistema.
- La **vista**, que és l'intermediari entre l'aplicació i l'usuari, és a dir, la qual cosa l'usuari veu en pantalla de l'aplicació. Per tant, la vista la compondran les diferents pàgines, formularis, etc, que l'aplicació mostrarà a l'usuari per a interactuar amb ell.
- El **controlador** (o controladors), que són els fragments de codi encarregats de coordinar el funcionament general de l'aplicació. Davant peticions dels usuaris, les recullen, les identifiquen, i accedeixen al model per a actualitzar o recuperar dades, i al seu torn, decideixen quina vista mostrar-li a l'usuari a continuació de l'acció que acaba de realitzar.

És un patró de disseny molt concís i ben estructurat, la qual cosa li ha valgut la fama que té hui dia. Entre els seus molts avantatges, permet aïllar el codi dels tres elements involucrats (vista, model i controlador), de manera que el treball és molt més modular i divisible, podent encarregar-se de les vistes, per exemple, un dissenyador web que no tinga molta idea de programació en el servidor, i del controlador un programador PHP que no tinga moltes nocions d'HTML.

En forma d'esquema, podríem representar-ho així:



5.2. Altres alternatives a MVC

Com a alternatives al patró MVC, i arran d'aquest mateix patró, van sorgir uns altres una mica més específics. Quasi tots ells tenen com a base la part del *model* (és a dir, l'accés i gestió de les dades o informació de l'aplicació) i la part de la *vista* (és a dir, la presentació a l'usuari). Així, podríem dir que l'únic punt "discordant" seria el controlador, que en altres patrons s'ha substituït per altres elements. De fet, al conjunt de patrons que segueixen aquesta filosofia (és a dir, centrar-se en la vista i el model, i afegir alguna cosa més), se'ls sol cridar en general MVW (en anglés, *Movel-View-Whatever*, en espanyol, *Model-Vista-Qualsevol cosa*). La finalitat d'això és, en alguns casos, descompondre el treball realitzat pels controladors en diversos submòduls, i en altres casos, prescindir directament del controlador. Vegem alguns dels patrons més populars.

El patró MVVM

El patró MVVM, com recullen les seues sigles, se centra exclusivament en els components del model i de la vista (*Model-Vista-Vista-Model*), i prescindeix del controlador. D'aquesta manera, l'usuari interactua directament amb la vista, i les accions o canvis que introduïsca en ella afecten directament el model, i viceversa (els canvis en el model es reflecteixen de manera automàtica en la vista).

Aquest patró està cobrant especial rellevància en les anomenades SPA (*Single Page Applications*), aplicacions web amb una sola pàgina que recarrega parcialment els seus continguts davant les accions de l'usuari. En aquests casos, no és necessari un controlador que diga quina vista carregar, perquè només hi ha una vista principal (que pot estar composta per subvistas), i si l'estructura és prou senzilla, vista i model poden estar intercomunicats sense intermediaris. Alguns frameworks sorgits últimament han donat encara més pes a aquest patró, com és el cas d'Angular.

El patró MOVE

El patró MOVE substitueix el controlador del patró MVC per dos elements. Un que denomina **operacions** (que seria l'O de les seues sigles), i que englobaria tot el conjunt d'accions que l'aplicació és capaç de realitzar, i un altre que serien els **esdeveniments** (que seria l'E de les seues sigles), i que representarien tots aquells successos que desencadenen que s'execute una operació determinada. Així, les accions dels usuaris són esdeveniments sobre l'aplicació que provoquen que s'executen determinades operacions. Aquestes operacions, al seu torn, poden accedir al model per a obtindre o actualitzar informació, i poden generar o cridar a una vista que mostrar a l'usuari com a resposta. Es divideix així la tasca dels controladors entre els esdeveniments i les operacions.

El patró MVP

El patró MVP substitueix el controlador (o controladors) del MVC pel que es denominen **presentadors**. Aquests presentadors són una espècie d'intermediaris entre el model i la vista, de manera que cada vista té el seu propi, i actua després de la vista per a comunicar-se amb el model, obtindre les dades, i carregar-los en ella per a mostrar-los a l'usuari. Es té així encapsulat amb cada vista el seu presentador, i l'aplicació pot considerar-se un conjunt de parells *vista-presentador*, que s'encarreguen de comunicar-se amb el model, que queda per darrere.

6. Recursos necessaris

Per a implantar una aplicació web i que els clients puguen utilitzar-la, necessitem comptar amb una sèrie recursos maquinari i programari.

- En el costat del **client**, simplement caldrà comptar amb un equip amb el maquinari necessari (depenent de l'aplicació web que siga, podrà ser un mòbil, una tauleta, portàtil, PC...), i, típicament, un navegador web instal·lat (encara que també podria tractar-se d'una aplicació híbrida per a mòbil, o d'escriptori, i en aquest cas faria falta l'aplicació en si).
- En el costat del **servidor**, normalment necessitarem almenys un PC servidor amb un maquinari relativament potent (quant a processador, memòria RAM i capacitat de disc dur). En ell, necessitarem tindre instal·lat:
- Un **servidor web**, o servidor d'aplicacions, on tindrem allotjada la nostra aplicació web, i atendrà les peticions dels clients. Normalment es tindrà un servidor web localitzable en Internet on instal·lar l'aplicació web definitiva (anomenat *servidor de producció*), i un altre en algun ordinador local on anar-la desenvolupant i provant fins a acabar-la (*servidor de proves*).
- Addicionalment, si la nostra aplicació web ho requereix, un **servidor de base de dades** o sistema gestor de bases de dades (SGBD). Aquesta opció és molt comuna en les aplicacions web, perquè la majoria accedeixen a una certa informació que, en general, està emmagatzemada en una base de dades.

6.1. Llenguatges

En parlar d'aplicacions web, és important determinar el llenguatge o llenguatges de programació en què es desenvolupen. En l'àmbit de les aplicacions web, distingim dos tipus de llenguatges:

- Llenguatges en l'entorn client o **llenguatges client**: són els que permeten que el client interactue amb l'aplicació web. Per a això, el client ha de poder veure l'aplicació web en el navegador, i interactuar amb ella (punxar en enllaços, emplenar formularis, etc.). En aquest costat, normalment es parla de **HTML** i **CSS** per al disseny de les pàgines (encara que no són llenguatges de programació pròpiament dits), i de **JavaScript** per a poder facilitar la interacció entre l'usuari i el navegador. També existeixen uns altres frameworks i eines que faciliten o amplien les possibilitats de desenvolupament en el client, com ara SASS (un compilador CSS que permet introduir una mica de programació en els documents CSS), i molts frameworks o llibreries JavaScript, com a Angular, React, Vue... A més, existeix algun llenguatge alternatiu, com **TypeScript**, similar a JavaScript però amb unes característiques més restrictives i estructurades quant a l'escriptura de codi.
- Llenguatges en l'entorn servidor o **llenguatges servidor**: són els que permeten que el servidor faça unes certes tasques quan li arriben les peticions dels clients, com per exemple consultar una base de dades, guardar informació en un fitxer, o carregar una foto que l'usuari està pujant al servidor. En aquest altre costat, existeixen diverses famílies de llenguatges que podem triar, depenent del servidor web que vulguem utilitzar. Per exemple, podem utilitzar llenguatge **ASP .NET** (per a servidors de Microsoft i entorns Windows), o el llenguatge **JSP** (llenguatge Java per a aplicacions web), o el llenguatge **PHP**, entre altres. També últimament s'ha fet un lloc en aquest grup el llenguatge **JavaScript**, a través del framework Node.js.

6.2. Exemples de programari

Hem vist a grans trets el maquinari i programari que necessitarem tant en els clients com en el servidor. Quin programari concret utilitzarem en aquest curs?

En el cas del **navegador** per al **client**, existeixen diverses opcions depenent del sistema operatiu del client: Mozilla Firefox, Google Chrome, Internet Explorer / Edge (només per a Windows), Safari (només en Windows i Macintosh), Opera... Possiblement els dos primers (Chrome i Firefox) són les millors opcions.

En el cas del **servidor web**, dependrà del tipus de llenguatge servidor que anem a utilitzar per a fer la nostra pàgina, i del sistema operatiu del servidor. En el nostre cas, emprarem llenguatge PHP, i també JavaScript amb el framework Node.js. Per al primer, emprarem el servidor **Apatxe**, que instal·larem a través d'algun sistema *XAMPP* o similar, que integra Apatxe amb un servidor MySQL/MariaDB i el llenguatge PHP preinstal·lats. Per al segon, instal·larem el framework **Node.js** per a construir amb ell el servidor, i alguns mòduls addicionals per a facilitar la tasca del desenvolupament de l'aplicació en si, com veurem en unitats posteriors.

Si emprarem llenguatge Java (JSP, servlets, etc.), podem instal·lar algun servidor senzill i gratuït com Tomcat, o servidors d'aplicacions més pesats i complexos com Glassfish. També funcionen en tota mena de sistemes Si optem per tecnologia .NET (ASP.NET), podem utilitzar IIS (Internet Information Services), un servidor web disponible per a Windows, ja que aquesta tecnologia només funciona en sistemes operatius Windows.

Per al **servidor de base de dades**, podem emprar diferents alternatives, com MySQL/MariaDB, PostreSQL, Oracle, o SQL Server. També podem optar per sistemes No-SQL, com MongoDB. Totes ofereixen versions gratuïtes (algunes d'elles amb recursos limitats per a ús personal) i comercials (més potents). MySQL, PostgreSQL i Oracle funcionen en diversos sistemes, però SQL Server és propietat de Microsoft, i funciona en sistemes Windows. En el nostre cas, optarem per un servidor **MySQL/MariaDB** per a la comunicació amb PHP i Apatxe (a través de l'esmentat XAMPP), i per un servidor **MongoDB** per a les aplicacions que fem amb Node.js.

IDEs

Per al desenvolupament d'aplicacions és necessari comptar amb un bon entorn de desenvolupament o IDE amb el qual editar el codi, depurar i provar les aplicacions.

Existeixen multitud d'opcions disponibles, la majoria d'elles gratuïtes. Des d'entorns de propòsit general vàlids per a molts llenguatges (Atom, Sublim, Visual Studio Code...) a uns altres més específics i orientats a algun llenguatge en concret, com PhpStorm.

En el nostre cas, optarem per **Visual Studio Code**, per la seua versatilitat i popularitat creixent. Permet instal·lar multitud d'extensions per a treballar amb diferents tipus de llenguatges i frameworks, i ens permet també una fàcil integració amb PHP i Node.js.

Alguns frameworks útils

També és convenient conéixer alguns frameworks populars i realment útils per al desenvolupament d'aplicacions web. Aquests frameworks es poden classificar depenent de si s'empren per a la part del client o la del servidor. En el primer cas, podem parlar d'Angular, React, Vue i alguns altres (encara que aquests són

ara com ara els més populars), però per a l'apartat que ens interessa, que són els frameworks en el costat del servidor, ens centrarem en:

- Un framework per a desenvolupament d'aplicacions PHP. En aquest apartat, els més rellevants són **Laravel** i **Symfony** hui dia.
- Un framework per al desenvolupament d'aplicacions JavaScript. Actualment només existeix el framework **Node.js** que, al seu torn, permet instal·lar altres frameworks sobre ell per a facilitar el desenvolupament d'aplicacions. Per exemple, utilitzarem el framework **Express**.

6.3. Webs d'interés

A continuació s'enumeren algunes webs on consultar o descarregar els recursos indicats en aquest tema, per al desenvolupament d'aplicacions en entorn servidor.

- XAMPP - https://www.apachefriends.org/es/index.html
- Apache - https://www.apache.org
- Node.js - https://nodejs.org
- MongoDB - https://www.mongodb.com
- Express - https://expressjs.com
- Laravel - https://laravel.com
- Symfony - https://symfony.com

> **Exercici 3:**
>
> Utilitza l'eina Google Trends per a buscar els termes de *Laravel*, *Symfony* i *Node.js*. Dedueix a partir de les cerques quin d'ells creus que és més popular actualment. Després, acudeix a la web de InfoJobs i busca ofertes de treball amb aquests tres frameworks, per a determinar quin és el més demandat en l'actualitat.