DESPLIEGUE DE APLICACIONES WEB

CFGS. Diseño de Aplicaciones Web (DAW). 18-19

IES FRANCESC DE BORJA MOLL

TEMA 5. SERVICIOS DE RED IMPLICADOS EN EL DESPLIEGUE DE UNA APLICACIÓN WEB.

ÍNDICE DE CONTENIDOS:

2. SERVICIO DE DIRECTORIO.

2.1. ¿Para qué usar un servicio de directorio?
2.2. Directorio vs DNS.
2.3. Organización del directorio LDAP.
2.4. Integración del servicio de directorio con otros servicios.
2.5. El formato de intercambio de datos LDIF.

3. VIRTUALIZACIÓN.

3.1. Contenedores
3.2. Virtualización

1. SERVIDORES DE NOMBRES DE DOMINIO.

Internet funciona mediante el protocolo **TCP/IP**, efectuando conexiones mediante IP. Por tanto, cada host en Internet se identifica mediante una **IP**, así es lo mismo visitar la página **http:/www.rediris.es** que **http://130.206.13.20**.

Entonces, sería posible visitar cada página web conociendo su IP, sólo que las personas están más acostumbradas a recordar nombres y no números. Por lo cual, debe existir una traducción de los nombres a IPs o viceversa. La operación de traducir la realiza el **servidor DNS** (*Domain Name System*) o un archivo de texto, típicamente denominando **hosts**, como el archivo /**etc**/**hosts** en sistemas GNU/Linux.

Pueden convivir en una misma máquina un servidor DNS y el archivo /etc/hosts, pero se ha de tener en cuenta la preferencia. Así, en caso de coexistir, primero se intentará la resolución IP/Nombre mediante el archivo /etc/hosts y, en caso de no encontrar correspondencia, actuará el servidor DNS.

El fichero /etc/hosts permite alias de nombres de dominios, esto es, una misma IP puede apuntar a nombres distintos. Cada línea del fichero comenzará con una IP y en la misma línea separada por espacios o tabuladores se pueden escribir los nombres de dominios correspondientes. El primer nombre, el más cercano a la IP, es considerado el principal, los demás son alias de éste.

1.1. Sistema de nombres de dominio.

¿Es necesario configurar un servidor DNS o se puede hacer la redirección mediante archivos de textos? Para la redirección deberá existir un **servidor DNS** que las resuelva o bien, en su defecto o a mayores, deberán existir las entradas correspondientes en el fichero del sistema local /**etc**/**hosts**. En caso de coexistir, primero se prueba la resolución en el fichero y luego en el servidor.

Entonces, ¿para qué montar un servidor si simplemente escribiendo en un fichero la relación IP/Nombre el sistema ya funcionaría? Pues, realmente depende, ya que si se está pensando en pocos equipos a resolver el nombre de dominio la simplicidad del fichero /**etc**/**hosts** permitiría no tener que montar un servidor, pero si el número de equipos que deben resolver el nombre en IP es elevado, el sistema del fichero es complicado de mantener y se debería pensar en montar un servidor DNS.

La complejidad radica en que en el fichero /etc/hosts los cambios son estáticos, así, para actualizar o activar un nuevo cambio debe editarse en todos los ficheros /**etc**/**hosts** implicados. Esto es, suponer que hay 20 equipos que quieren resolver una página web, por ejemplo *www.debian.org* el procedimiento sería aproximadamente el siguiente:

1. Se escribe la página web en cada equipo en la barra de direcciones del navegador.
2. Se traduce el nombre DNS a una IP, o bien mediante servidores DNS, o bien mediante ficheros estáticos /etc/hosts, con lo cual se debe modificar este fichero en cada cliente.

Pero, ¿y si la resolución tiene lugar mediante servidores DNS?, ¿y por qué servidores DNS y no servidor DNS? Existe, a modo de resumen, un procedimiento de resolución DNS, más o menos, similar al siguiente –se encontrará el procedimiento exacto más adelante-:

- Primero, se debe averiguar que servidor DNS resuelve el domino raíz 'org' a una IP.
- Segundo, una vez obtenida esa IP que gobierna el dominio raíz 'org', se le pregunta por la IP del servidor DNS que gobierna el subdominio 'debian' bajo 'org'.
- Tercero, una vez obtenida la IP del servidor DNS que gobierna el dominio 'debian.org' se le pregunta por la IP del equipo 'www.debian.org'

Entonces: ¿cuántos servidores DNS existen a la hora de preguntar? ¿existe un número limitado de redirecciones de consultas? ¿y, si se vuelve a hacer la misma consulta, hay que repetir el proceso?. No existe un número limitado de redirección de consultas, lo que sucede es que las consultas se van escalando hasta encontrar un servidor DNS que las resuelva, y escalando y escalando puede ser que las consultas se resuelvan en los últimos servidores DNS a los cuales se puede preguntar: los servidores raíz.

Pero, puede ser que no sea necesario escalar las consultas, puesto que todos los servidores DNS son servidores caché, lo que significa que recuerdan las consultas efectuadas. Por lo tanto, si se hace una consulta que ya está guardada en la caché, la respuesta es casi instantánea y ya ha sido resuelta. Es más, los equipos clientes, desde donde se hace la consulta a través del navegador como se indicaba en el ejemplo, también poseen una memoria caché DNS, de tal forma que anteriormente a preguntar al servidor DNS, se mira en la caché del propio sistema operativo, y si se obtiene la respuesta el proceso se ha acabado.

El sistema DNS en realidad es una base de datos distribuida, que permite la administración local de segmentos que juntos componen toda la base de datos local. Los datos de cada segmento están disponibles para toda la red a través de un esquema cliente-servidor jerárquico.

1.1.1. ¿Cómo es un nombre de dominio?

Normalmente en al barra de direcciones del navegador se suele escribir algo similar a: www.debian.org. Entonces, vienen siendo unos caracteres separados por puntos. ¿Qué es lo que significan esos puntos? ¿Qué dividen? Además, en el ejemplo expuesto, al escribir *www.debian.org* el navegador autocomplementa esta petición a *http://www.debian.org*, ¿por qué?

- Primero: Los puntos separan dominios y subdominios, empezando de derecha a izquierda se tendrán dominios de primer nivel y dominios de segundo, tercer, …, n-ésimo nivel, denominados subdominios. Así:
    - **org** es el dominio de primer nivel que identifica a organizaciones.
    - **debian** es un subdominio, en este caso dominio de segundo nivel bajo org, que identifica al nombre de la organización o al nombre de la empresa, sucursal, etc.
    - **www** es un subdominio, en este caso dominio de tercer nivel bajo debian, que identifica al equipo donde está colgada la página web, esto es, identifica el servidor web que aloja la página web. Es el dominio www que el servidor DNS redirecciona a la IP del servidor web.
- Segundo: http:// es el protocolo de hypertexto que permite la correcta visualización de la página web en el navegador. Es lo que el navegador autocomplementa en caso de no estipular uno propio en la barra de direcciones URL con en nombre de dominio.

Los dominios de primer nivel identifican el tipo de página web que se solicita o bien la localización de la misma, por ejemplo:

- **net** identifica redes.
- **com** identifica comercio.
- **es** identifica localización España.

Esto suele ser lo común, más no es obligatorio, es decir, si una empresa posee un dominio **com** puede dedicarse al sector de redes de comunicaciones y no poseer el dominio **net**, así como puede ser una empresa localizada en España y no poseer el dominio **es**.

A nivel gramatical los dominios deben cumplir una serie de requisitos. Por ejemplo:

- Sólo pueden estar compuestos de letras (alfabeto inglés), números y guiones ("-").
- No pueden empezar o terminar por guiones.
- Tienen que tener menos de 63 caracteres sin incluir la extensión, y más de uno o dos dependiendo del dominio de primer nivel.

Ahora bien, hoy día ya es posible registrar dominios con caracteres de otras lenguas no inglesas, como la ñ o la ç. Estos dominios se denominan **multilingües**.

1.1.2. Jerarquías de nombres de dominio.

El espacio de nombres de dominio está organizado de forma jerárquica. El nivel más alto en la jerarquía es el dominio raíz, que se representa como un punto (".") y el siguiente nivel en la jerarquía se llama dominio de nivel superior (TLD Top Level Domain). Sólo hay un dominio raíz, pero hay muchos TLDs y cada TLD se llama dominio secundario del dominio raíz. En este contexto, el dominio raíz es el dominio principal, ya que está un nivel por encima de un TLD y cada TLD, a su vez, pueden tener muchos dominios hijos. Los hijos de los dominios de nivel superior se llaman de segundo nivel, los del segundo nivel se llaman de tercer nivel, los del tercer nivel de cuarto, y así sucesivamente.

Por lo tanto el DNS, organiza los nombres de máquina (hostname) en una jerarquía de dominios separados por el carácter punto '.'. Un **dominio** es una colección de nodos relacionados de alguna forma -porque están en la misma red, tal como los nodos de una empresa-. Por ejemplo:

```
rrhh.departamento.empresa.org
marketing.departamento.empresa.org
contabilidad.consultas.empresa.org
```

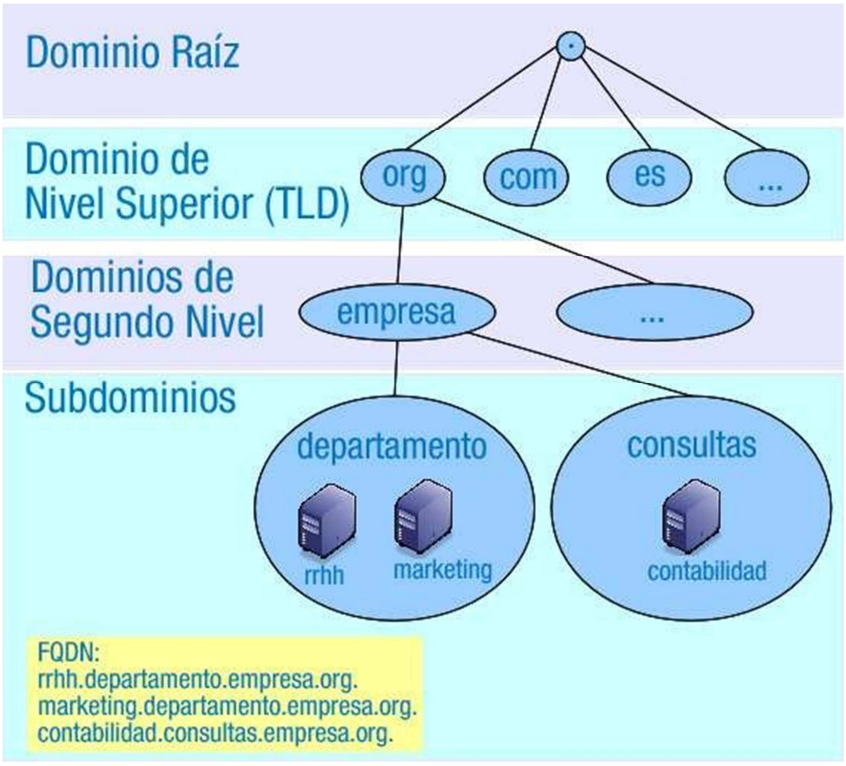
Donde:

- La empresa agrupa sus nodos en el dominio de primer nivel "org". Éste es un TLD.
- La empresa tiene un subdominio, dominio de segundo nivel "empresa" bajo "org". Así "empresa" es un dominio de segundo nivel, hijo del TLD "org".
- A su vez se pueden encontrar nuevos subdominios dentro, en este caso: "departamento" y "consultas". Es decir, dominios de tercer nivel, hijos a su vez del dominio de segundo nivel "empresa".
- Finalmente, un nodo que tendrá un nombre completo conocido como totalmente cualificado o FQDN (*fully qualified domain name*), que es la concatenación de: TLD, dominio de segundo nivel, dominio de tercer nivel, etc., tal como:

```
rrhh.departamento.empresa.org,      marketing.departamento.empresa.org,
contabilidad.consultas.empresa.org.
```

También es posible tener un dominio de cuarto nivel, dominio de quinto nivel, y así sucesivamente.

En la siguiente figura se puede ver una parte del espacio de nombres. La raíz del árbol, que se identifica con un punto sencillo, es lo que se denomina dominio raíz y es el origen de todos los dominios. Para indicar que un nombre es FQDN, a veces se termina su escritura en un punto, aunque por lo general se omite. Este punto significa que el último componente del nombre es el dominio raíz.



Así, por ejemplo en el nombre de dominio:

El símbolo del dominio raíz es el punto situado más a la derecha del nombre del dominio.

Sólo hay una raíz de dominio, pero hay más de 250 dominios de nivel superior, clasificados en los siguientes tres tipos:

- TLD de código de país (ccTLD): dominios asociados con países y territorios. Hay más de 240 ccTLD. Están formados por 2 letras, por ejemplo: es, uk, en, y jp.
- Dominios de nivel superior genéricos (gTLD): están formados por 3 o más letras. A su vez se subdividen en:
    - Dominios de internet patrocinados (sTLD): representan una comunidad de intereses, es decir, detrás existe una organización u organismo público que propone el dominio y establece las reglas para optar a dicho dominio. Por ejemplo: edu, gov, int, mil, aero, museum.

- Dominios de internet no patrocinados (uTLD). Sin una organización detrás que establezca las reglas para optar a dicho dominio. La lista de gTLD incluye: com, net, org, biz,

1.2. Ventajas del DNS.

¿Qué pasaría si se dispone de 20 equipos y en todos se actualiza una entrada DNS en el fichero /etc/hosts, salvo en 3 de ellos? Esos tres quedarían no actualizados. ¿Y si en la próxima actualización el cambio no se replica en otros 3, que pueden ser los mismos o no? ¿Y en la próxima …?

El sistema de modificar el archivo /etc/hosts no parece buena idea, puesto que al ser cambios estáticos, más de un cambio puede quedar en el tintero, obteniendo al final un sistema no homogéneo. Así, parece claro que la solución, para obtener un sistema no heterogéneo es el DNS.

El DNS permite que cualquier cambio efectuado solamente en un servidor se replique en todos los servidores DNS que la configuración permita, de tal forma que el cambio sólo se efectúa en un servidor, obteniendo así facilidad y simplicidad en el cambio. Por lo tanto, cualquier cambio es dinámico: se configura solamente un servidor y éste se encarga de replicar el cambio.

Por otro lado, qué es lo que pasa si un servidor DNS está caído y por lo tanto la conectividad con el mismo no es posible: ¿quedaría todo el sistema inhabilitado? ¿se podría conectar aún a páginas web? Bien, pues como cada servidor DNS se ocupa de su zona, eso no imposibilita el acceso a otras zonas y por lo tanto a la visibilidad y conectividad de otros dominios que no dependan de ese servidor DNS. Es más, es posible que no solamente exista un servidor DNS configurado para controlar esa zona, y por lo tanto tampoco esa zona estuviese no visible.

Una zona DNS es aquella parte del DNS para la cual se ha delegado la administración, es decir, cuando se configura un dominio en un servidor DNS, éste debe pertenecer a una zona. Así, en los archivos de configuración de zona se indicará qué IP va con el servicio web www, el servicio de correo mail, etc.

Los tipos de zonas posibles son dos:

1. **Zona de Búsqueda Directa:** las resoluciones de esta zona devuelven la dirección IP correspondiente al recurso solicitado. Realiza las resoluciones que esperan como respuesta la dirección IP de un determinado recurso.

2. **Zona de Búsqueda Inversa:** las resoluciones de esta zona buscan un nombre de equipo en función de su dirección IP; una búsqueda inversa tiene forma de pregunta, del estilo "¿Cuál es el nombre DNS del equipo que utiliza la dirección IP 192.168.200.100?".

Los servidores DNS no solamente sirven para la resolución de nombres en Internet, también se pueden utilizar en redes locales. Así, las entradas existentes en los DNS de la red local podrían ser visibles en Internet, o no, solamente sirviendo resolución a los equipos de la red local. De esta forma, cuando un usuario de la red local intenta acceder a un recurso local, podrá utilizar **nombres** en lugar de direcciones IP. Si el usuario desea acceder fuera de la red local a algún recurso en Internet, el DNS local nunca podrá llevar a cabo dicha resolución y se la traslada al siguiente servidor DNS (que sí estará en Internet) en su jerarquía de servidores DNS, hasta que la petición sea satisfecha.

Por ejemplo, con un servidor DNS en red local, que resuelve la IP 192.168.200.100 a cliente.local y viceversa, se puede ejecutar el comando ping indistintamente contra dicha IP o contra el nombre del equipo en el dominio:

```
ping 192.168.200.100
ping cliente.local
```

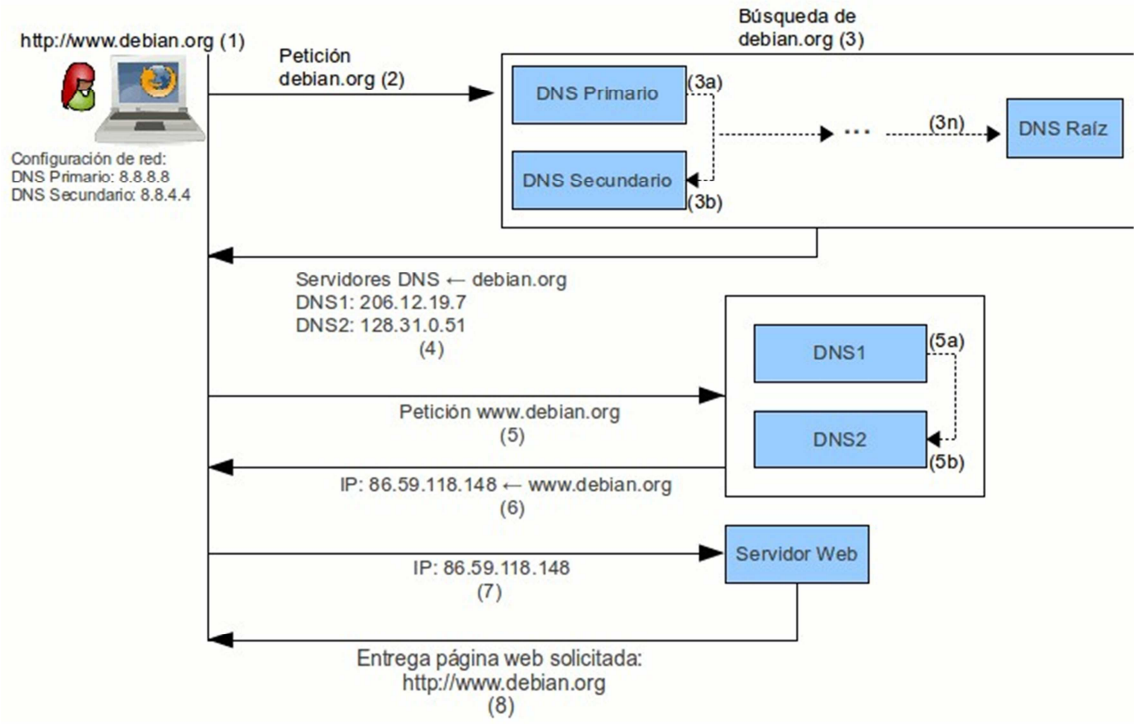
En ambos casos, se debería obtener la misma respuesta. Esto suele ser muy útil cuando los hosts reciben su IP por DHCP ya que puede ocurrir que se desconozca la IP que tiene cierto equipo pero sí conocer su nombre en el dominio, que será invariable.

Se pueden resumir las ventajas de la configuración y empleo de un servidor DNS en las siguientes:

1. Desaparece la carga excesiva en la red y en los hosts: ahora la información esta distribuida por toda la red, al tratarse de una base de datos distribuida.
2. No hay duplicidad de nombres: el problema se elimina debido a la existencia de dominios controlados por un único administrador. Puede haber nombres iguales pero en dominios diferentes.
3. Consistencia de la información: ahora la información que está distribuida es actualizada automáticamente sin intervención de ningún administrador.

1.3. Funcionamiento del DNS.

La siguiente imagen representa el funcionamiento del DNS, tomando como ejemplo la página web *www.debian.org* y considerando que la información de la petición del dominio a buscar no se encuentra en tu ordenador o en un servidor DNS local existente en tu red o en tu ordenador.



1. A través del navegador se quiere consultar la página web oficial de Debian escribiendo en la barra de direcciones la URL *http://www.debian.org.*
2. El navegador busca la información de las DNS del dominio **debian.org**.
3. Internet está ordenada en forma de árbol invertido, si no encuentra la información en el ordenador, irá a buscarla a los servidores DNS que se posee en la configuración de red del ordenador, típicamente los proporcionados por el Proveedor de Servicios a Internet (ISP): DNS Primario (3a) o DNS Secundario (3b). De no estar, seguirá buscándola a niveles superiores y, en último lugar, lo encontrará en el Servidor de Nombres Raíz: DNS Raíz (3n).
4. La información buscada: las IP correspondientes al servidor DNS que gobierna el dominio debian.org, llega al ordenador: DNS1→ 206.12.19.7 y DNS2→ 128.31.0.51. Suelen ser dos porque las especificaciones de diseño de DNS recomiendan que, como mínimo, deben existir dos servidores DNS para alojar cada zona, a la que pertenece cada dominio.
5. EL ordenador ahora intentará conectar con el servidor DNS1 (5a) o ante cualquier problema de conexión con éste lo intentará con el servidor DNS2 (5b). Éstos son los servidores de nombres donde se encuentra información acerca de dónde se puede buscar la página web (servidor de la web), una dirección de correo electrónico (servidor de correo), etc.
6. EL ordenador recibirá la información acerca de la localización de la página web, o sea, la dirección IP del servidor web donde está alojada la página.
7. El ordenador se dirigirá luego al servidor web y buscará la página web en él.

8. Por último, el servidor web devuelve la información pedida y tú recibes la página web, visualizándola en el navegador.

Pero, si se vuelve a consultar la página web oficial de Debian escribiendo en la barra de direcciones la URL *http://www.debian.org*, se podría repetir de nuevo todo el proceso, pero para ello hay que establecer dos situaciones:

1. El host desde el que se vuelve a realizar la consulta es el mismo: Si no lo es, antes de repetir todo el proceso se intentaría con lo expuesto en el siguiente punto, pero si es el mismo, al haber hecho la consulta desde este host, la resolución dominio-IP se guarda durante algún tiempo en la memoria caché del mismo, por lo cual no será necesario repetir todo el proceso de nuevo. Si el tiempo en el que la memoria caché guarda la resolución ha expirado se volverá a repetir el proceso de nuevo.
2. Existe un servidor DNS caché en tu red o en tu host: por lo tanto, si un segundo cliente, que tiene configurado este servidor DNS, vuelve a realizar la misma petición, como este servidor tiene la respuesta almacenada en su memoria caché, responderá inmediatamente sin tener que cursar la petición a ningún servidor DNS de Internet. Si el tiempo en el que la memoria caché guarda la resolución ha expirado se volverá a repetir el proceso de nuevo.

1.4. DNS dinámico.

¿Es posible si se dispone de una conexión a Internet con IP dinámica ofrecer servicios en Internet?

Parece claro que si se dispone de una IP estática de conexión a Internet, previo pago de un plus por disponer siempre de una misma IP para la conexión a Internet, simplemente se debería enrutar las peticiones de los servicios que se ofrece a los hosts que esperan la conexión a esos servicios. Si además, se posee nombres de dominios, se puede redireccionar esos nombres a las IP de los hosts a través del servidor DNS.

Volviendo a la pregunta anterior ¿qué pasaría si se quisiera hacer lo mismo y no se dispone de IP estática? esto es, cada vez que se conecta a Internet la IP, aunque a veces sea la misma, no siempre es la misma. Entonces, es posible ofrecer servicios con IP dinámica, pero se necesitaría:

1. Recoger la IP de la conexión cada vez que se te conecta a Internet.
2. Una vez recogida la IP difundirla en Internet. Para difundirla, o bien se hace de forma estática, y cada vez que se recoge se realizan los cambios necesarios para difundirla, o bien de forma dinámica se configura un programa para que automáticamente recoja la IP y la difunda.

Está claro, que la mejor opción es difundirla de forma dinámica, para ello se puede hacer uso de servicios ofrecidos, incluso de forma gratuita, por **DynDNS**, **No-IP** y **FreeDNS**. De hecho, hoy en día, los routers que los ISP suelen montar ya poseen la opción de configuración por DNS dinámica.

Entonces, el **DNS dinámico** es un sistema que permite la actualización en tiempo real de la información sobre nombres de dominio situados en un servidor de nombres, siendo usado, mayoritariamente, para asignar un nombre de dominio de Internet a un ordenador con dirección IP variable (dinámica).

El DNS dinámico, así, puede ofrecer servicios en Internet en hosts que posean conexión con dirección IP dinámica, la típica configuración que los ISP ofrecen para conectarse a Internet.

De todos modos, aunque existe la posibilidad de ofrecer servicios en Internet desde tu propia casa, se debe tener en cuenta que, usualmente, la infraestructura técnica y la electrónica de red que se posea no se puede comparar con los servidores ofrecidos por empresas de Hosting (*balanceadores de carga, redundancia en caso de fallos, generadores eléctricos que garanticen conexión eléctrica permanente a pesar de caída eléctrica, ancho de banda necesario para permitir múltiples conexiones concurrentes sin perjudicar el servicio ofrecido, …etc*).

Si no se tiene configurado un servidor DNS con las entradas de dominio necesarias, se puede generar estas entradas modificando el archivo /etc/hosts, añadiéndolas al final del mismo:

- #IP nombre-dominio
- 192.168.200.250 empresa1.com www.empresa1.com
- 192.168.200.250 empresa2.com www.empresa2.com

Cada campo, de cada entrada, puede ir separado por espacios o por tabulados.

Estas entradas solamente serán efectivas en el equipo en el que se desee modificar el archivo /etc/hosts. Así, se debe modificar el archivo /etc/hosts en cada equipo que se quiera que se resuelvan esas entradas.

1.5. Tipos de servidores DNS.

Existen varios tipos de servidores DNS.

Dependiendo de la configuración y funcionamiento de los servidores, éstos pueden desempeñar distintos papeles:

- **Servidores primarios** (primary name servers). Estos servidores almacenan la información de su zona en una base de datos local. Son los responsables de mantener la información actualizada y cualquier cambio debe ser notificado a este servidor.
- **Servidores secundarios** (secundary name servers). También denominados *esclavos*, aunque a su vez pueden ser *maestros* de otros servidores secundarios. Son aquellos que obtienen los datos de su zona desde otro servidor que tenga autoridad para esa zona. El proceso de copia de la información se denomina *transferencia de zona*.
- **Servidores maestros** (master name servers). Los servidores maestros son los que transfieren las zonas a los servidores secundarios. Cuando un servidor secundario arranca busca un servidor maestro y realiza la transferencia de zona. Un servidor maestro para una zona puede ser a la vez un servidor primario o secundario de esa zona. Así, se evita que los servidores secundarios sobrecarguen al servidor primario con transferencias de zonas. Por ejemplo, en la imagen el servidor DNS3 pide la zona al servidor DNS2 y no al servidor DNS1, con lo cual se evita la sobrecarga del servidor DNS1. Los servidores maestros extraen la información desde el servidor primario de la zona.
- **Servidores sólo caché** (caching-only servers). Los servidores sólo caché no tienen autoridad sobre ningún dominio: se limitan a contactar con otros servidores para resolver las peticiones de los clientes DNS. Estos servidores mantienen una memoria caché con las últimas preguntas contestadas. Cada vez que un cliente DNS le formula una pregunta, primero consulta en su memoria caché. Si encuentra la dirección IP solicitada, se la devuelve al cliente; si no, consulta a otros servidores, apunta la respuesta en su memoria caché y le comunica la respuesta al cliente. Disponer de un servidor caché DNS en nuestra red local aumenta la velocidad de la conexión a Internet pues cuando se navega por diferentes lugares, continuamente se están realizando peticiones DNS. Si nuestro caché DNS almacena la gran mayoría de peticiones que se realizan desde la red local, las respuestas de los clientes se satisfarán prácticamente de forma instantánea proporcionando al usuario una sensación de velocidad en la conexión. Muchos routers ADSL ofrecen ya este servicio de caché, tan solo hay que activarlo y configurar una o dos IPs de servidores DNS en Internet. En los equipos de red local se podrían poner como DNS primario la IP de nuestro router y como DNS secundario una IP de un DNS de Internet.

Los servidores secundarios son importantes por varios motivos. En primer lugar, por seguridad: debido a que la información se mantiene de forma redundante en varios servidores a la vez. Si un servidor tiene problemas, la información se podrá recuperar desde otro. Y en segundo lugar, por velocidad: porque evita la sobrecarga del servidor principal distribuyendo el trabajo entre

distintos servidores situados estratégicamente (por zonas geográficas, por ejemplo).

Todos los servidores DNS guardan en la caché las consultas que resolvieron.

Una transferencia de zona puede darse en cualquiera de los casos siguientes:

- Cuando vence el intervalo de actualización de una zona.
- Cuando un servidor maestro notifica los cambios de la zona a un servidor secundario.
- Cuando se inicia el servicio Servidor DNS en un servidor secundario de la zona.
- Cuando se utiliza el comando rndc en un servidor secundario de la zona para iniciar manualmente una transferencia desde su servidor maestro, por ejemplo:

  rndc retransfer proyecto-empresa.local

  donde: retransfer → indica que la acción a realizar es una transferencia.

  proyecto-empresa.local → es el nombre de la zona que quieres transferir.

1.6. Servidores raíz.

La organización que gestiona globalmente los servidores raíz por concesión del gobierno estadounidense es la ICANN (*Internet Corporation for Assigned Names and Numbers*), la cual es una organización sin fines de lucro que opera a nivel internacional, responsable de asignar espacio de direcciones numéricas de protocolo de Internet (IP), identificadores de protocolo y de las funciones de gestión [o administración] del sistema de nombres de dominio de primer nivel genéricos (gTLD) y de códigos de países (ccTLD), así como de la administración del sistema de servidores raíz. Aunque en un principio estos servicios los desempeñaba IANA y otras entidades bajo contrato con el gobierno de EE.UU., actualmente son responsabilidad de ICANN.

ICANN es responsable de la coordinación de la administración de los elementos técnicos del DNS para garantizar una resolución unívoca de los nombres, de manera que los usuarios de Internet puedan encontrar todas las direcciones válidas. Para ello, se encarga de supervisar la distribución de los identificadores técnicos únicos usados en las operaciones de Internet, y delegar los nombres de dominios de primer nivel, como: *com, info,* etc.

Otros asuntos que preocupan a los usuarios de Internet, como reglamentación para transacciones financieras, control del contenido de Internet, correo electrónico de publicidad no solicitada (SPAM) y protección de datos, están fuera del alcance de la misión de coordinación técnica de ICANN.

Si se desea conocer más sobre ICANN:

https://archive.icann.org/tr/spanish.html

Las empresas, ciudades u organizaciones podrán registrar sus propios dominios genéricos, tras la decisión adoptada el 20 de Junio de 2011 por la ICANN en Singapur. Esta iniciativa permitirá que las direcciones de los dominios puedan terminar con el nombre de compañía, ciudad, etc., en vez de *.com, .net* o *.org*.

Los servidores raíz son entidades distintas. Hay 13 servidores raíz o, más precisamente, 13 direcciones IP en Internet en las que pueden encontrarse a los servidores raíz (los servidores que tienen una de las 13 direcciones IP pueden encontrarse en docenas de ubicaciones físicas distintas). Todos estos servidores almacenan una copia del mismo archivo que actúa como índice principal de las agendas de direcciones de Internet. Enumeran una dirección para cada dominio de nivel principal (*.com, .es*, etc.) en la que puede encontrase la propia agenda de direcciones de ese registro.

En realidad, los servidores raíz no se consultan con mucha frecuencia (considerando el tamaño de Internet) porque una vez que los ordenadores de la red conocen la dirección de un dominio de nivel principal concreto pueden conservarla, y sólo comprueban de forma ocasional que esa dirección no haya cambiado. Sin embargo, los servidores raíz siguen siendo una parte vital para el buen funcionamiento de Internet.

Las entidades encargadas de operar los servidores raíz son bastante autónomas pero, al mismo tiempo, colaboran entre sí y con ICANN para asegurar que el sistema permanece actualizado con los avances y cambios de Internet.

Los trece servidores raíz DNS se denominan por las primeras trece letras del alfabeto latino, de la A hasta la M (A.ROOT-SERVERS.NET., B.ROOT-SERVERS.NET., …, M.ROOT-SERVERS.NET.), y están en manos de 9 organismos y corporaciones diferentes e independientes, principalmente universidades, empresas privadas y organismos relacionados con el ejercito de EE.UU. Aproximadamente la mitad depende de organizaciones públicas estadounidenses.

Si se desea consultar una lista actualizada de servidores raíz DNS:

http://www.internic.net/zones/named.root

1.7. Tipos de registros DNS.

Una base de datos DNS se compone de uno o varios archivos de zonas utilizados por el servidor DNS. Cada zona mantiene un conjunto de registros de recursos estructurados.

Todos los registros de recursos (RR) tienen un formato definido que utiliza los mismos campos de nivel superior, según se describe en la tabla siguiente:

Formato de los registros de recursos DNS:

Campo	Descripción
Propietario	Indica el nombre de dominio DNS que posee un registro de recursos. Este nombre es el mismo que el del nodo del árbol de la consola donde se encuentra un registro de recursos.
Tiempo de vida (TTL)	Para la mayor parte de los registros de recursos, este campo es opcional. Indica el espacio de tiempo utilizado por otros servidores DNS para determinar cuánto tarda la información en caché en caducar un registro y descartarlo. Por ejemplo, la mayor parte de los registros de recursos que crea el servicio del servidor DNS heredan el TTL mínimo (predeterminado) de 1 hora desde el registro de recurso de inicio de autoridad (SOA) que evita que otros servidores DNS almacenen en caché durante demasiado tiempo. En un registro de recursos individual, puede especificar un TTL específico para el registro que suplante el TTL mínimo (predeterminado) heredado del registro de recursos de inicio de autoridad. También se puede utilizar el valor cero (0) para el TTL en los registros de recursos que contengan datos volátiles que no estén en la memoria caché para su uso posterior una vez se complete la consulta DNS en curso.
Clase	Contiene texto nemotécnico estándar que indica la clase del registro de recursos. Por ejemplo, el valor "IN" indica que el registro de recursos pertenece a la clase Internet. Este campo es *obligatorio*.
Tipo	Contiene texto nemotécnico estándar que indica el tipo de registro de recursos. Por ejemplo, el texto nemotécnico "A" indica que el registro de recursos almacena información de direcciones de host. Este campo es *obligatorio*.
Datos específicos del registro	Un campo de longitud variable y *obligatorio* con información que describe el recurso. El formato de esta información varía según el tipo y clase del registro de recursos.

En la siguiente tabla se muestran los registros DNS más utilizados:

Nota: en los siguientes ejemplos de registros de recurso, el campo TTL se omite en caso de ser opcional. El campo TTL se ha incluido en la sintaxis de cada registro para indicar dónde puede agregarse.

Tipos de registros DNS:

Registro	Descripción, sintaxis y ejemplo
A	**Descripción**: Address (Dirección). Este registro se usa para traducir nombres de hosts a direcciones IP versión 4. **Sintaxis**: *propietario clase ttl* **A** *IP_version4*. **Ejemplo**: **host1.ejemplo.com IN A 127.0.0.1**.
AAAA	**Descripción**: Address (Dirección). Este registro se usa para traducir nombres de hosts a direcciones IP versión 6. **Sintaxis**: *propietario clase ttl* **AAAA** *IP_version6*. **Ejemplo**: **host1ipv6.ejemplo.com. IN AAAA 1234:0:1:2:3:4:567:89ab**.
CNAME	**Descripción**: Canonical Name (Nombre Canónico). Se usa para crear nombres de hosts adicionales, o alias. Hay que tener en cuenta que el nombre de host al que el alias referencia debe haber sido definido previamente como registro tipo "A". Comúnmente usado cuando un servidor con una sola dirección IP ejecuta varios servicios, como: ftp, web... y cada servicio tiene su propia entrada DNS. También es utilizado cuando el servidor web aloja distintos dominios en una misma IP (virtualhosts). **Sintaxis**: *propietario ttl clase* **CNAME** *nombreCanónico*. **Ejemplo**: **nombrealias.ejemplo.com CNAME nombreverdadero.ejemplo.com**. Como se ha comentado anteriormente **nombreverdadero.ejemplo.com** previamente debe estar definido como registro tipo **A**.
NS	**Descripción**: Name Server (Servidor de Nombres). Indica qué servidores de nombres tienen total autoridad sobre un dominio concreto. Cada dominio se puede asociar a una cantidad cualquiera de servidores de nombres. **Sintaxis**: *propietario ttl IN* **NS** *nombreServidorNombreDominio*. **Ejemplo**: **ejemplo.com. IN NS nombreservidor1.ejemplo.com**.
MX	**Descripción**: Mail eXchange (Registro de Intercambio de Correo). Asocia un nombre de dominio a una lista de servidores de intercambio de correo para ese dominio. **Sintaxis**: *propietario ttl clase* **MX** *preferencia hostIntercambiadorDeCorreo*.

Registro	Descripción, sintaxis y ejemplo

Ejemplo: **ejemplo.com. MX 10 servidorcorreo1.ejemplo.com**.

El número, en este caso 10, indica la preferencia, y tiene sentido en caso de existir varios servidores de correo. A menor número mayor preferencia.

PTR

Descripción: Pointer (Indicador). Traduce direcciones IP en nombres de dominio. También conocido como 'registro inverso', ya que funciona a la inversa del registro "A".

Sintaxis: *propietario ttl clase* **PTR** *nombreDominioDestino*.

Ejemplo: **1.0.0.10.in-addr.arpa. PTR host.ejemplo.com.**

SOA

Descripción: Start Of Authority (Autoridad de la zona). Proporciona información sobre el servidor DNS primario de la zona.

Sintaxis: *propietario clase* **SOA** *servidorNombres personaResponsable (numeroSerie intervaloActualización intervaloReintento caducidad tiempoDeVidaMínimo)*.

Ejemplo:

```
@ IN SOA nombreServidor.ejemplo.com. postmaster.ejemplo.com. (
 1 ; número de serie
 3600 ; actualizar [1h]
 600 ; reintentar [10m]
 86400 ; caducar [1d]
 3600 ) ; TTL mínimo [1h]
```

El propietario (servidor DNS principal) se especifica como "@" porque el nombre de dominio es el mismo que el origen de todos los datos de la zona (ejemplo.com). Se trata de una convención de nomenclatura estándar para registros de recursos y se utiliza más a menudo en los registros SOA. El número de serie es el número de versión de esta base de datos. Debes incrementar este número cada vez que modificas la base de datos.

TXT

Descripción: TeXT (Información textual). Permite a los dominios identificarse de modos arbitrarios.

Sintaxis: *propietario ttl clase* **TXT** *cadenaDeTexto*.

Ejemplo: **ejemplo.com. TXT "Ejemplo de información de nombre de dominio adicional."**

SPF

Descripción: Sender Policy Framework. Es un registro de tipo TXT que va creado en una zona directa del DNS, en la cual se pone las informaciones del propio servidor de correo con la sintaxis SPF. Se utiliza para evitar el envío de correos suplantando identidades. Por lo tanto, ayuda a combatir el SPAM, ya que, en este registro se

Registro	Descripción, sintaxis y ejemplo

especifica cual o cuales hosts están autorizados a enviar correo desde el dominio dado. El servidor que recibe, consulta el S para comparar la IP desde la cual le llega, con los datos de este registro.

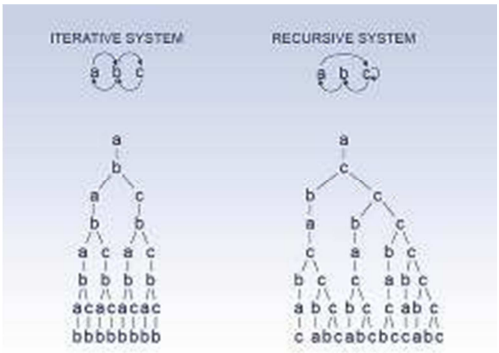
Sintaxis: *propietario ttl clase* **IN SPF** *cadenaDeTexto*.

Ejemplo: **ejemplo.com IN SPF "v=spf1 a:mail.ejemplo.com -all"**.

1.8. Funcionamiento del cliente DNS.

Cuando se utiliza en un programa un nombre DNS, éste debe ser resuelto a una IP. Entonces, un cliente DNS busca el nombre que se utiliza en el programa, consultando los servidores DNS para resolver el nombre. Cada mensaje de consulta que envía el cliente contiene tres grupos de información, que especifican una pregunta que tiene que responder el servidor:

- Un nombre de dominio DNS especificado, indicado como un nombre de dominio completo (FQDN *fully qualified domain name*).
- Un tipo de consulta especificado, que puede establecer un registro de recursos por tipo o un tipo especializado de operación de consulta.
- Una clase especificada para el nombre de dominio DNS.



Por ejemplo, el nombre especificado puede ser el nombre completo de un equipo, como **rrhh.departamento.empresa.org.**, y el tipo de consulta especificado para buscar un registro de recursos de dirección (A) por ese nombre. Considerar una consulta DNS como una pregunta de un cliente a un servidor en dos partes, como: "¿Tiene algún registro de recursos de dirección (A) de un equipo llamado '**rrhh.departamento.empresa.org.**'?". Cuando el cliente recibe una respuesta del servidor, lee e interpreta el registro de recursos "A" respondido, y aprende la dirección IP del equipo al que preguntó por el nombre.

Las consultas DNS se resuelven de diferentes formas:

- A veces, un cliente responde a una consulta localmente mediante la información almacenada en la caché obtenida de una consulta anterior.

- El servidor DNS puede utilizar su propia caché de información de registros de recursos para responder a una consulta.
- Un servidor DNS también puede consultar, o ponerse en contacto con otros servidores DNS, en nombre del cliente solicitante para resolver el nombre por completo y, a continuación, enviar una respuesta al cliente. Este proceso se llama **recursividad**.
- Además, el mismo cliente puede intentar ponerse en contacto con servidores DNS adicionales para resolver un nombre. Cuando un cliente lo hace, utiliza consultas adicionales e independientes en función de respuestas de referencia de los servidores. Este proceso se llama **iteración**.

En general, el proceso de consulta DNS se realiza en dos partes:

- La consulta de un nombre comienza en un equipo cliente y se pasa al solucionador (resolver), el servicio Cliente DNS, para proceder a su resolución.
- Cuando la consulta no se puede resolver localmente, se puede consultar a los servidores DNS según sea necesario para resolver el nombre.

1.8.1. Consultas recursivas.

Tu ordenador (cliente DNS) formula una **consulta** a tu servidor DNS preferido (el que se tiene configurado como primero en la configuración de red, generalmente el proveedor de Internet). Cuando el servidor DNS recibe una consulta, primero comprueba si puede responder la consulta en las zonas configuradas localmente en el servidor, esto es, en las zonas que posee autoridad. Así, pueden ocurrir dos situaciones:

1. Si el nombre consultado existe, esto es, coincide con un registro de recursos correspondiente en la información de zona local, el servidor responde con autoridad y usa esta información para resolver el nombre consultado.

2. Si el nombre consultado no existe, esto es, no existe ninguna información de zona para el nombre consultado, a continuación el servidor comprueba si puede resolver el nombre mediante la información almacenada en la caché local de consultas anteriores. De nuevo, se dan dos situaciones:
    a. Si el servidor preferido puede responder al cliente solicitante con una respuesta coincidente de su caché, finaliza la consulta y responde con esta información.
    b. Si el servidor preferido no puede responder al cliente solicitante con una respuesta coincidente de su caché, el proceso de consulta puede continuar y se usa la recursividad para resolver completamente el nombre. Esto implica la asistencia de otros

servidores DNS para ayudar a resolver el nombre. De forma predeterminada, el servicio cliente DNS solicita al servidor que utilice un proceso de recursividad para resolver completamente los nombres en nombre del cliente antes de devolver una respuesta. En la mayor parte de los casos, el servidor DNS se configura, de forma predeterminada, para admitir el proceso de recursividad como se muestra en el gráfico siguiente.

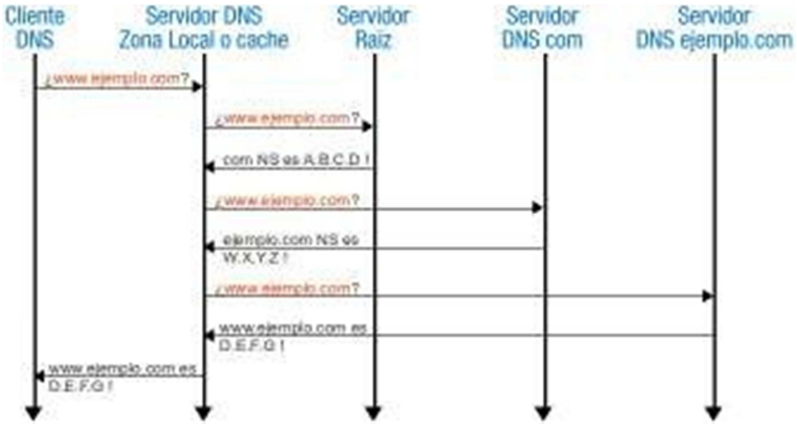


Diagrama que demuestra el funcionamiento de las consultas recursivas. El cliente solamente se conecta con un servidor DNS y este realiza todas las consultas necesarias para devolverle al cliente la información pedida. Aparecen 5 líneas verticales que finalizan en punta de flecha simbolizando el transcurrir del tiempo. Cada línea respectivamente tiene un encabezado, de izquierda a derecha: cliente DNS, Servidor DNS Zona Local o Caché, Servidor Raíz, Servidor DNS com, Servidor DNS ejemplo.com.

De arriba hacia abajo y de izquierda a derecha aparecen líneas horizontales, terminadas en puntas de flecha que indican el sentido de la consulta y que demuestran el ciclo de vida en una consulta recursiva, a saber:

- *El cliente DNS pregunta por la IP de **www.ejemplo.com** al servidor DNS Zona Local o caché.*
- *Éste a su vez pregunta al servidor Raíz, quien le devuelve la IP del servidor que gobierna el dominio com.*
- *Ahora el servidor DNS pregunta a ese servidor por la IP de **www.ejemplo.com** y éste le devuelve la IP del servidor que gobierna ejemplo.com*
- *Y así hasta que el servidor DNS tiene la IP.*
- *Una vez tiene la IP se la comunica al cliente.*

Para que el servidor DNS realice la recursividad correctamente, primero necesita información de contacto útil acerca de los otros servidores DNS del espacio de nombres de dominio DNS. Esta información se proporciona en forma de *sugerencias de raíz*, una lista de los registros de recursos preliminares que puede utilizar el servicio DNS para localizar otros servidores DNS que tienen autoridad para la raíz del árbol del espacio de nombres de dominio DNS. Los servidores raíz tienen autoridad para el dominio raíz y los dominios de nivel superior en el árbol del espacio de nombres de dominio DNS.

Un servidor DNS puede completar el uso de la recursividad utilizando las sugerencias de raíz para encontrar los servidores raíz. En teoría, este proceso permite a un servidor DNS localizar los servidores que tienen autoridad para cualquier otro nombre de dominio DNS que se utiliza en cualquier nivel del árbol del espacio de nombres.

Por ejemplo, pensar en la posibilidad de usar el proceso de recursividad para localizar el nombre "www.ejemplo.com" cuando el cliente consulte un único servidor DNS. El proceso ocurre cuando un servidor y un cliente DNS se inician y no tienen información almacenada en la caché local disponible para ayudar a resolver la consulta de un nombre. El servidor supone que el nombre consultado por el cliente es para un nombre de dominio del que el servidor no tiene conocimiento local, según sus zonas configuradas.

Primero, el servidor preferido analiza el nombre completo y determina que necesita la ubicación del servidor con autoridad para el dominio de nivel superior "com". A continuación, utiliza una consulta iterativa al servidor DNS "com" para obtener una referencia al servidor "ejemplo.com". Finalmente, se entra en contacto con el servidor "ejemplo.com". Ya que este servidor contiene el nombre consultado como parte de sus zonas configuradas, responde con autoridad al servidor original que inició la recursividad. Cuando el servidor original recibe la respuesta que indica que se obtuvo una respuesta con autoridad a la consulta solicitada, reenvía esta respuesta al cliente solicitante y se completa el proceso de consulta recursiva.

Aunque el proceso de consulta recursiva puede usar muchos recursos cuando se realiza como se describe anteriormente, tiene algunas ventajas en el rendimiento para el servidor DNS. Por ejemplo, durante el proceso de recursividad, el servidor DNS que realiza la búsqueda recursiva obtiene información acerca del espacio de nombres de dominio DNS. Esta información se almacena en la caché del servidor y se puede utilizar de nuevo para ayudar a acelerar la obtención de respuestas a consultas subsiguientes que la utilizan o concuerdan con ella. Con el tiempo, esta información almacenada en caché puede crecer hasta ocupar una parte significativa de los recursos de memoria del servidor, aunque se limpia siempre que el servicio DNS se activa y desactiva.

1.8.2. Consultas iterativas.

La iteración es el tipo de resolución de nombres que se utiliza entre clientes y servidores DNS cuando se dan las condiciones siguientes:

- El cliente solicita el uso de la recursividad, pero ésta se encuentra deshabilitada en el servidor DNS.
- El cliente no solicita el uso de la recursividad cuando consulta el servidor DNS.

Una solicitud iterativa de un cliente informa al servidor DNS de que el cliente espera la mejor respuesta que el servidor DNS pueda proporcionar inmediatamente, sin entrar en contacto con otros servidores DNS.

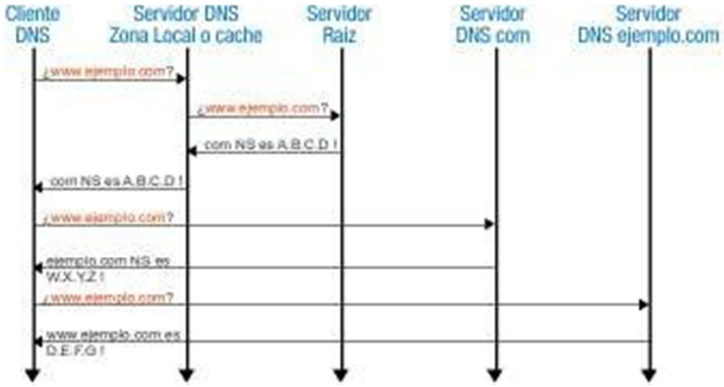


Diagrama que demuestra el funcionamiento de las consultas iterativas. El cliente realiza todas las consultas necesarias hasta obtener la información solicitada.

Aparecen 5 lineas verticales que finalizan en punta de flecha simbolizando el transcurrir del tiempo. Cada línea respectivamente tiene un encabezado, de izquierda a derecha: Cliente DNS, Servidor DNS Zona Local o Caché, Servidor Raíz, Servidor DNS com, Servidor DNS ejemplo.com.

De arriba hacia abajo y de izquierda a derecha aparecen líneas horizontales, terminadas en puntas de flecha que indican el sentido de la consulta y que demuestran el ciclo de vida en una consulta recursiva, a saber:

- *El cliente DNS pregunta por la IP de www.ejemplo.com al servidor DNS Zona Local o caché.*
- *Éste a su vez pregunta al servidor Raíz, quien le devuelve la IP del servidor que gobierna el dominio com.*
- *Ahora es de nuevo el cliente DNS quien sigue buscando la IP y continúa preguntando a todos los servidores necesarios (en el diagrama todos) hasta llegar a la resolución de la misma.*

Cuando se utiliza la iteración, un servidor DNS responde al cliente en función de su propio conocimiento específico acerca del espacio de nombres, sin tener en cuenta los datos de los nombres que se están consultando. Por ejemplo, si un servidor DNS de una intranet recibe una consulta de un cliente local para "**www.ejemplo.com**", es posible que devuelva una respuesta de su caché de nombres. Si el nombre consultado no está almacenado actualmente en la caché de nombres del servidor, puede que, para responder, el servidor proporcione una referencia, es decir, una lista de registros de recursos de dirección (A) y de servidor de nombres (NS) para otros servidores DNS que estén más cerca del nombre consultado por el cliente.

Cuando se proporciona una referencia, el cliente DNS asume la responsabilidad de continuar efectuando consultas iterativas a otros servidores DNS configurados para resolver el nombre. Por ejemplo, en el caso más complicado, el cliente DNS puede expandir su búsqueda a los servidores de dominio raíz en Internet en un esfuerzo por localizar los servidores DNS que tienen autoridad para el dominio "com". Una vez en contacto con los servidores raíz de Internet, puede recibir más respuestas iterativas de estos servidores DNS que señalan a los servidores DNS de Internet reales para el dominio "ejemplo.com". Cuando se proporcionan registros de estos servidores DNS al cliente, éste puede enviar otra consulta iterativa a los servidores DNS externos del dominio ejemplo en Internet, que pueden responder con una respuesta definitiva y con autoridad.

Cuando se utiliza la iteración, un servidor DNS puede ayudar en la resolución de la consulta de un nombre además de devolver su mejor respuesta propia al cliente. En la mayor parte de las consultas iterativas, un cliente utiliza su lista de servidores DNS configurada localmente para entrar en contacto con otros servidores de nombres a través del espacio de nombres DNS si su servidor DNS principal no puede resolver la consulta.

1.8.3. Consultas inversas.

En la mayoría de las consultas DNS, los clientes normalmente realizan una búsqueda directa. Este tipo de consulta espera recibir una dirección IP como respuesta a la consulta. Pero, DNS también proporciona un proceso de búsqueda inversa, es decir, buscar un nombre de host a través de una dirección IP. Así, una búsqueda inversa busca la respuesta a una pregunta tipo como la siguiente: ¿Cuál es el nombre DNS del host que utiliza la dirección IP 192.168.200.100?.

DNS no se diseñó originalmente para aceptar este tipo de consulta. Un problema de compatibilidad con el proceso de consulta inversa es la diferencia en la forma en que el espacio de nombres DNS organiza e indexa los nombres, y cómo se asignan las direcciones IP. Si el único método para responder a la pregunta anterior fuera buscar en todos los dominios del espacio de nombres DNS, una consulta inversa llevaría demasiado tiempo y requeriría un procesamiento demasiado largo como para ser útil.

Entonces, para resolver este problema, en el estándar DNS se definió y se reservó un dominio especial para las IP versión 4, el dominio **in-addr.arpa**, en el espacio de nombres DNS de Internet con el fin de proporcionar una forma práctica y confiable para realizar las consultas inversas. Al crear el espacio de nombres inverso, los subdominios del dominio **in-addr.arpa** se crean con el orden inverso de los números en la notación decimal con puntos de las direcciones IP. Por ejemplo, para la IP 192.168.200.100 su resolución inversa sería:

```
100.200.168.192.in-addr.arpa.
```

Este orden inverso de los dominios para el valor de cada octeto es necesario porque, a diferencia de los nombres DNS, cuando se leen las direcciones IP de izquierda a derecha se interpretan al contrario. Cuando se lee una dirección IP de izquierda a derecha, se ve desde su información más general (una dirección IP de red) en la primera parte de la dirección a la información más específica (una dirección IP de host) que contienen los últimos octetos. Por esta razón, se debe invertir el orden de los octetos de las direcciones IP cuando se crea el árbol del dominio **in-addr.arpa.**.

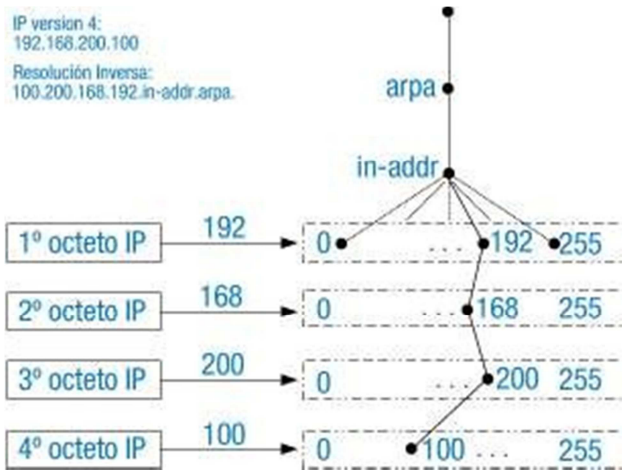


Diagrama que representa una IP versión 4 en su resolución inversa en el dominio in-addr.arpa. Así se ve arriba a la izquierda el texto: IP versión 4: 192.168.200.100. Resolución inversa: 100.200.168.192.in-addr.arpa. A la derecha, aparece de arriba hacia abajo representado y unidos mediante líneas:

- *Un punto.*
- *El dominio arpa.*
- *A continuación in-addr.*
- *A continuación .192 dentro de una caja entre 0 y 255.*
- *A continuación .168 dentro de una caja entre 0 y 255.*
- *A continuación .200 dentro de una caja entre 0 y 255.*
- *A continuación .100 dentro de una caja entre 0 y 255.*

Cada caja, como ser ve en la figura representa un octecto de la IP versión 4.

Finalmente, el árbol del dominio **in-addr.arpa**, tal como se crea en DNS, requiere que se defina un tipo de registro de recursos adicional: el registro de recursos de puntero (PTR). Este registro de recursos se utiliza para crear una asignación en la zona de búsqueda inversa que, normalmente, corresponde a un registro de recurso de dirección (A) de host con nombre para el nombre del equipo DNS de un host en su zona de búsqueda directa.

El dominio **in-addr.arpa** se usa en todas las redes TCP/IP que se basan en el direccionamiento del Protocolo de Internet versión 4 (IPv4). Para el Protocolo

de Internet versión 6 (IPv6) se usa un nombre de dominio especial diferente, el dominio **ip6.arpa.**

Se ha de tener en cuenta que, si el servidor DNS no puede responder el nombre de la consulta inversa, se puede utilizar la resolución DNS normal (ya sea la recursividad o la iteración) para localizar un servidor DNS con autoridad para la zona de búsqueda inversa y que contenga el nombre consultado. En este sentido, el proceso de resolución de nombres utilizado en una búsqueda inversa es idéntico al de una búsqueda directa.

1.9. Cómo funcionan los DNS preferidos y alternativos.

El servidor DNS preferido es aquel con el que el cliente prueba en primer lugar. También es el servidor en el que el cliente DNS actualiza sus registros de recursos. Si el servidor DNS preferido falla, el cliente prueba con el servidor DNS alternativo.

Opcionalmente, se puede especificar una lista completa de servidores DNS alternativos. Los servidores DNS preferidos y alternativos especificados se consultan en el orden que aparezcan en la lista.

Sin un servidor DNS preferido, el cliente DNS no puede consultar un servidor DNS. Sin un DNS alternativo, las consultas no se resolverán si el servidor DNS preferido falla.

Los pasos siguientes indican el proceso para entrar en contacto con servidores DNS preferidos y alternativos:

1. El servidor DNS preferido responde primero a una consulta DNS o a una actualización DNS.
2. Si el servidor DNS preferido no responde a una consulta DNS o a una actualización DNS, la consulta o actualización se redirige al servidor DNS alternativo.
3. Si el servidor DNS alternativo no responde y el cliente DNS está configurado con las direcciones IP adicionales de servidores DNS, el cliente DNS envía la consulta o actualización al siguiente servidor DNS de la lista.
4. Si alguno de los servidores DNS (un servidor preferido, un servidor alternativo o cualquier otro de la lista) no responde, dicho servidor se quita temporalmente de la lista.
5. Si ninguno de los servidores DNS responden, la consulta o actualización del cliente DNS no se realiza.

En los equipos tipo GNU/Linux se puede configurar estos servidores en el archivo **/etc/resolv.conf** e incluso se puede realizar balanceo de carga entre

ellos, así como la modificación del tiempo de espera efectuado desde que un servidor falla hasta que se prueba con otro.

La configuración sería:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

En este caso si 8.8.8.8 falla la resolución se realizará a través de 8.8.4.4. El problema es que por defecto el valor de tiempo de espera (timeout) asignado es 5 segundos, por lo que tardará un tiempo en detectar que tiene que utilizar el segundo DNS y todo irá muy lento. Para solucionarlo, se tiene que usar la directiva "options" y modificar el timeout. Así, se puede poner 1 segundo como se demuestra en el siguiente ejemplo:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
options timeout:1
```

Otra opción también interesante es "rotate", que permite distribuir la carga entre todos los servidores listados y evitar que todas las peticiones vayan siempre al primero:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
options timeout:1 rotate attempts:1
```

Configurando estas opciones se asegura que en caso del fallo del servidor DNS preferido el rendimiento de la máquina no se degrade.

Se ha de tener en cuenta que es posible y más que probable que el fichero **/etc/resolv.conf** sea modificado cuando se configura la red mediante un gestor de conexión de redes, como: NetworkManager, wicd **...** Por lo tanto, se ha de revisar este fichero.

1.10. Comandos (I).

A la hora de saber si se tiene conectividad con alguna máquina en Internet, o en red local, se suele utilizar el comando **ping**, el cual indica, según su respuesta, si se posee conectividad con la máquina en cuestión. El comando **ping** se puede utilizar para consultar direcciones IP o nombres de dominios.

Por lo tanto, el comando **ping** debe ser capaz de consultar información sobre el sistema de nombres de dominio; es un resolutor, un programa cliente capaz de consultar información sobre el sistema de nombres de dominio. Normalmente, un resolutor trabaja discretamente en segundo plano y los usuarios no conocen su presencia, es decir, que toda consulta de un cliente DNS a su servidor suele realizarla el programa que se invoca (ping, ftp, telnet,

mail, navegador web, etc.). Por ejemplo, si solicitas una conexión ftp a **ftp.rediris.es**, la aplicación ftp que se emplee llama a un programa resolutor local que busca la dirección IP de ese ordenador **130.206.1.5** sin que se tenga conciencia de ello, esto es, para el usuario el proceso es transparente. Además de este trabajo en segundo plano, el usuario puede conectarse directamente al programa resolutor enviando consultas y resolviendo respuestas. Comandos resolutores típicos en sistemas operativos GNU/Linux son: **nslookup**, **host** y **dig**.

El comando **nslookup**, en algunas distribuciones GNU/Linux ya no está soportado pues está obsoleto (deprecated). Por lo tanto, hoy en día, se suelen utilizar el comando **host** para consulta de direcciones IP y el comando **dig** para consulta de servidores DNS activos. A continuación, se muestra cómo funcionan estos comandos:

Ejemplos de resolución directa: Resolución de nombre a IP.

1. **Comando *nslookup*:**

   Para consultar la dirección IP del ordenador ftp.rediris.es**,** basta con ejecutar:

```
nslookup ftp.rediris.es
alumno@servidor-fp:~$ nslookup ftp.rediris.es
Server:          8.8.8.8
Address:     8.8.8.8#53

Non-authoritative answer:
ftp.rediris.es     canonical name = zeppo.rediris.es.
Name:     zeppo.rediris.es
Address: 130.206.1.5
```

   Donde se puede ver que *ftp.rediris.es* es un alias de *zeppo.rediris.es* cuya dirección IP es 130.206.1.5

2. **Comando *host*:**

   Para consultar la dirección IP del ordenador *ftp.rediris.es*, basta con ejecutar:

```
host ftp.rediris.es
alumno@servidor-ftp:~$ host ftp.rediris.es
ftp.rediris.es is an alias for zeppo.rediris.es.
zeppo.rediris.es has address 130.206.1.5
```

   Donde se puede ver que *ftp.rediris.es* es un alias de *zeppo.rediris.es* cuya dirección IP es 130.206.1.5

3. **Comando *dig*:**

   Para consultar la dirección IP del ordenador *ftp.rediris.es*, basta con ejecutar:

```
dig ftp.rediris.es
alumno@servidor-ftp:~$ dig ftp.rediris.es

; <<>> DiG 9.7.3 <<>> ftp.rediris.es
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
31214
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;ftp.rediris.es.                IN      A

;; ANSWER SECTION:
ftp.rediris.es.              7200       IN       CNAME
zeppo.rediris.es.
zeppo.rediris.es.          5195          IN          A
130.206.1.5

;; Query time: 76 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Fri Jul 29 11:13:44 2011
;; MSG SIZE rcvd: 68
```

   Donde se puede ver que *ftp.rediris.es* es un alias de *zeppo.rediris.es* cuya dirección IP es 130.206.1.5

1.10.1. Comandos (II).

Ejemplos de resolución inversa: Resolución de IP a nombre.

1. **Comando *nslookup*:**

   Para consultar el nombre de la IP 130.206.1.5, basta con ejecutar:

```
nslookup 130.206.1.5
alumno@servidor-fp:~$ nslookup 130.206.1.5
Server:          80.58.61.254
Address:     80.58.61.254#53

Non-authoritative answer:
5.1.206.130.in-addr.arpa     name = zeppo.rediris.es.
```

```
Authoritative answers can be found from:
```

Donde se puede ver que la IP 130.206.1.5 corresponde con el nombre de dominio *zeppo.rediris.es*.

2. Comando *host*:

Para consultar el nombre de la IP 130.206.1.5, basta con ejecutar:

```
host 130.206.1.5
     alumno@servidor-ftp:~$ host 130.206.1.5
5.1.206.130.in-addr.arpa    domain    name    pointer
zeppo.rediris.es.
```

Donde se puede ver que la IP 130.206.1.5 corresponde con el nombre de dominio *zeppo.rediris.es*.

3. Comando *dig*:

Para consultar el nombre de la IP 130.206.1.5, basta con ejecutar:

```
dig -x 130.206.1.5
alumno@servidor-fp:~$ dig -x 130.206.1.5

; <<>> DiG 9.7.3 <<>> -x 130.206.1.5
;; global options: +cmd
;; Got answer:
;;  ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
38384
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;5.1.206.130.in-addr.arpa.     IN     PTR

;; ANSWER SECTION:
5.1.206.130.in-addr.arpa.  7200         IN         PTR
zeppo.rediris.es.

;; Query time: 73 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Fri Jul 29 12:03:30 2011
;; MSG SIZE rcvd: 72
```

Donde se puede ver que la IP 130.206.1.5 corresponde con el nombre de dominio *zeppo.rediris.es*, e incluso el registro de recursos empleado: **PTR** *(puntero para asignar un nombre de dominio DNS inverso basado en la dirección IP de un equipo que señala al nombre de dominio DNS directo de ese equipo)*.

2. SERVICIO DE DIRECTORIO.

Un **servicio de directorio** (**SD**) es una aplicación o un conjunto de aplicaciones que almacena y organiza la información sobre los usuarios de una red de ordenadores, sobre recursos de red, y permite a los administradores gestionar el acceso de usuarios a los recursos sobre dicha red. Además, los servicios de directorio actúan como una capa de abstracción entre los usuarios y los recursos compartidos.

Seguramente se habrá utilizado más de una vez algún tipo de directorio o servicio de directorio, como por ejemplo: una guía telefónica impresa en papel o una revista con la programación televisiva.

Los directorios, por lo tanto, permiten localizar información y para ello definen qué información se almacenará y en qué modo.

Los directorios anteriormente comentados presentan una serie de problemas, en contra de los directorios electrónicos, a saber:

1. Son estáticos:

    1. Cuando se busca un teléfono en la guía telefónica, la información más actualizada es la de la fecha de edición impresa de la guía. Esto quiere decir que si una persona modifica sus datos o da de alta una nueva línea no aparecerán los cambios hasta la próxima edición impresa.

    2. En el caso de la programación televisiva, el tiempo de renovación de la información se reduce, posiblemente sea semanalmente. Pero, ¿y si existe algún cambio de última hora en el medio de la semana? La información también quedaría obsoleta.

Por contra, los directorios electrónicos pueden ser consultados/actualizados en tiempo real, por lo que su fiabilidad es mucho mayor.

2. Son inflexibles: en el contenido y en su organización:

    1. ¿Qué pasaría si en la guía telefónica se quisiera introducir una nueva información sobre el propietario de un teléfono? Está claro que la visualización del nuevo contenido no es instantáneo, habría que modificarlo y el usuario debería esperar a la nueva edición.
    2. ¿Qué pasaría si en la programación televisiva se quisiera incorporar un nuevo logotipo de una cadena de televisión? Pues, lo mismo que lo comentado anteriormente.

       Por contra, los directorios electrónicos pueden modificar cualquier contenido y éste se verá reflejado al instante.
    3. ¿Qué pasaría si se quisiera buscar en la guía telefónica un teléfono por la calle donde vive el usuario?

4. ¿Qué pasaría si en la programación televisiva se quisiera buscar todas las películas sobre acción que se emiten a una determinada hora, pero solamente los días a la semana que te interese?

Puede ser que se encuentre esa información pero, desde luego, no es muy flexible la búsqueda de la misma. Por contra, los directorios electrónicos permiten que la búsqueda de información sea localizada de distintas maneras, gracias a cómo está organizada.

3. Son inseguros: dificultad para controlar el acceso a la información.

1. ¿Cómo se impide que un usuario no pueda buscar un teléfono en la guía telefónica?
2. ¿Cómo se impide que un menor pueda leer cierto contenido sobre, por ejemplo, un programa no educativo?

Los directorios electrónicos sí permiten controlar el acceso a la información: solamente aquel que disponga de las claves de acceso obtendrá la información.

4. Difícilmente configurable:

1. ¿Cómo hacer en la guía telefónica para realizar una búsqueda solamente sobre un segundo apellido, de una zona urbana y con teléfonos que poseen dos números que se determinen?
2. ¿Cómo hacer en la programación televisiva para realizar una búsqueda sobre cadenas por satélite para Europa que emitan en franjas horarias determinadas deportes y, a poder ser, torneos o ligas profesionales?

Bien, parece ser que manejar tanta cantidad de información para ser ofrecida en ese tipo de búsquedas la hace no impresa e incluso inmanejable. Por contra, los directorios electrónicos pueden establecer la información que recibe una persona en función de sus necesidades.

2.1. ¿Para qué usar un servicio de directorio?

Por lo visto anteriormente los directorios electrónicos permiten, de forma eficiente:

1. Encontrar información:

   Los directorios electrónicos a diferencia de los clásicos permiten acceder a la información contenida en los mismos de múltiples formas. Así,

comparando con la guía telefónica tradicional, un directorio electrónico permite realizar búsquedas, no solamente por orden alfabético, sino también por: apellido: dirección, teléfono… ¿Cómo se realizaría una búsqueda por teléfono en una guía telefónica tradicional?

Es más, se podría sumar campos de búsqueda, como por ejemplo: dirección y apellido.

2. Gestionar información:

   En los directorios electrónicos pueden existir varios usuarios que en tiempo real estén realizando modificaciones, como agregar/editar/eliminar distintos usuarios con sus correspondientes campos. Además, esta información ya estaría visible para todas aquellas aplicaciones que accedan a la misma. Centralizar así los datos en un directorio evita tener que sincronizar varios directorios, con el consiguiente riesgo que esto provoca, pues: ¿qué pasaría si la sincronización no tuvo lugar y una aplicación accede a los datos? Pues que obtendría los datos no actualizados, o error en los mismos.

   Un caso muy común es el de los servidores Web con autenticación: si solamente se dispone de un servidor web la solución es sencilla, puesto que solamente se necesitaría actualizar una base de datos de usuarios, pero ¿y si se dispone de más de un servidor web que debe acceder a la misma base de datos? Entonces, la cosa se complica, puesto que se debe sincronizar a los distintos servidores. Es más, y si esa base de datos se quisiera aprovechar para ofrecer otro servicio distinto del de los servidores web? Pues, todo el trabajo no sería aprovechable, y por lo tanto sería mejor desde un principio adaptar este sistema a los servicios de directorios.

3. Control de seguridad:

   Los servicios de directorios no simplemente permiten delimitar el acceso a los usuarios, sino que también proporcionan una solución al problema de gestión de certificados digitales. Así, permiten:

   1. Su creación: Incorporar a los certificados los datos contenidos en el directorio.
   2. Su distribución: Tener accesibles mediante un protocolo estándar los certificados.
   3. Su destrucción: Revocar los certificados de forma sencilla simplemente borrando el certificado del directorio.
   4. Su ubicación: Los usuarios pueden acceder a través del directorio a los certificados de los restantes usuarios, de forma muy sencilla y fácil de integrar con las aplicaciones.

Por todo ello, las aplicaciones prácticas que poseen los servicios de directorio son muy diversas y ventajosas, como por ejemplo: autenticación de usuarios: en aplicaciones web, correo electrónico, ..., sistemas de control de entradas a edificios, bases de datos comunes en organizaciones, en sistemas operativos: gestión de cuentas de acceso, servidores de certificados, libretas de direcciones compartidas...

2.2. Directorio vs DNS.

Tanto un servicio de directorio como un servicio DNS proporcionan acceso a una base de datos jerárquica, pero difieren en:

1. Los servidores de directorio no están particularizados a una acción concreta, sino orientados de forma más general, mientras que el servicio DNS está dedicado a la traducción de nombres de dominios a direcciones IP.
2. La información almacenada en el servicio de directorio no es fija, mientras que en el servicio DNS tiene una estructura fija.
3. El servicio de directorio permite actualizaciones, mientras que el servicio DNS no las permite.
4. Los servicios de directorio suelen utilizar protocolos orientados a conexión (TCP), mientras que el servicio DNS opera con protocolos no orientados a conexión (UDP).

Pero, no por ello, poseen el impedimento de trabajar juntos, es más, usualmente se encontrarán unidos de la mano en aplicaciones web con distintas funcionalidades, como: servidores de correo, gestión de proyectos e incidencias, etc. Así, suele ser necesario acceder a las URL de las aplicaciones web mediante nombres de dominio DNS y una vez en ellas autenticarse por medio de LDAP (*Lightweight Directory Access Protocol*, en español Protocolo Ligero de Acceso a Directorios).

LDAP es un protocolo que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también se considera una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

Antes de intentar configurar una aplicación web con autenticación LDAP se debería probar la instalación, configuración y autenticación por medio de una base de datos SQL.

2.3. Organización del directorio LDAP.

El servicio de directorio puede estar centralizado o distribuido:

- **Centralizado**: En este caso un único servidor ofrece todo el servicio de directorio respondiendo a todas las consultas de los clientes.

- **Distribuido**: Si el directorio está distribuido, varios servidores proporcionan el servicio de directorio. Cuando está distribuido, los datos pueden estar fraccionados y/o replicados:

    - Cuando está fraccionada, cada servidor de directorio almacena un subconjunto único y no solapado de la información, es decir, una entrada es almacenada en un solo servidor.
    - Cuando la información está replicada, una entrada puede estar almacenada en varios servidores.

  Generalmente, cuando el servicio de directorio es distribuido, parte de la información está fraccionada y parte está replicada.

En 1988, la CCITT (ahora ITU-T Internacional Telecomunication Union) creó el estándar X.500 sobre servicios de directorio, el cual organiza las entradas en el directorio de manera jerárquica, capaz de almacenar gran cantidad de datos, con grandes capacidades de búsqueda y fácilmente escalable. X.500 especifica que la comunicación entre el cliente y el servidor de directorio debe emplear el protocolo DAP, pero DAP es un protocolo a nivel de aplicación, por lo que, tanto al cliente como el servidor debían implementar completamente la torre de protocolos OSI.

LDAP surge como una alternativa a DAP. Las claves del éxito de LDAP en comparación con DAP de X.500 son:

- El modelo funcional de LDAP es más simple y ha eliminado opciones raramente utilizadas en X.500, siendo más fácil de comprender e implementar.
- LDAP representa la información mediante cadenas de caracteres en lugar de complicadas estructuras ASN.1 (*Abstract Syntax Notation One*).

El directorio LDAP tiene una estructura en forma de árbol denominado **DIT** *(Directory Information Tree)*. Cada entrada del directorio describe un **objeto**: persona, impresora, etc. La ruta completa a una entrada la identifica de modo inequívoco y se conoce como **DN** *(Distinguished Name)* y está compuesto por una secuencia de partes más pequeñas llamadas **RDN** (*Relative Distinguished Name*), de forma similar a como el nombre de un fichero consiste en un camino de directorios en muchos sistemas operativos.

Una **clase de objeto (objectClass)** es una descripción general de un tipo de objeto. Todos los objetos de LDAP deben tener el atributo **objectClass**. La definición de **objectClass** especifica qué atributos requiere un objeto LDAP, así como las clases de objetos que pueden existir. Los valores de este atributo los pueden modificar los clientes, pero el atributo **objectClass** en sí no puede eliminarse.

Un **esquema (schema)** define: qué clases de objetos se pueden almacenar en el directorio, qué atributos deben contener, qué atributos son opcionales y el formato de los atributos.

Por lo general, existen dos tipos de objetos:

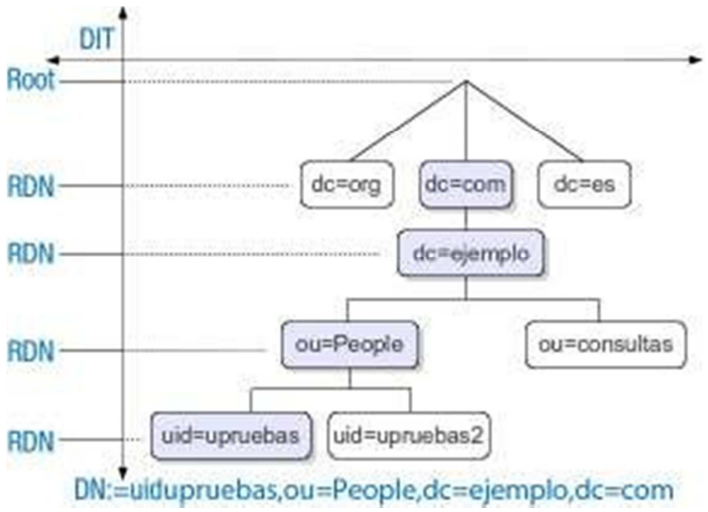
- **Contenedor**: Este tipo de objeto puede contener a su vez otros objetos. Algunos ejemplos de estos elementos son: **Root** (elemento raíz del árbol de directorios que no existe en realidad), **c** (country), **ou** (OrganizationalUnit) y **dc** (domainComponent).

  La figura análoga al contenedor es el directorio (carpeta) de un sistema de archivos.

- **Hoja**: Este tipo de objeto se encuentra al final de una rama y carece de objetos subordinados. Algunos ejemplos son: Person/InetOrgPerson o groupofNames.

En la cúspide de la jerarquía del directorio se encuentra el elemento raíz **Root**. A este elemento le puede seguir en un nivel inferior **c** (country), **dc** (domainComponent) ó **o** (organization).

La siguiente imagen ilustra las relaciones jerárquicas dentro de un árbol de directorios LDAP:



La figura representa un DIT ficticio con entradas en cuatro niveles. Cada entrada se corresponde con una casilla en la figura. En este caso, el nombre válido completo DN del empleado ficticio **upruebas** es:

```
dn: uid=upruebas,ou=People,dc=ejemplo,dc=com
```

La definición global de qué tipo de objetos han de guardarse en el DIT se realiza mediante un *esquema*. El tipo de objeto se determina mediante la *clase de objeto*. La clase de objeto especifica qué atributos *deben* o *pueden* ser asignados a un objeto determinado. Por lo tanto, un esquema debe contener

definiciones de todas las clases de objetos y atributos que van a utilizarse en el escenario de aplicación. Existen algunos esquemas de uso extendido (**RFC 2252** y **2256**). No obstante, si el entorno en el que va a utilizarse el servidor LDAP lo requiere, también pueden crearse nuevos esquemas en función del usuario o pueden combinarse varios esquemas entre sí.

2.4. Integración del servicio de directorio con otros servicios.

De lo expuesto anteriormente puede deducirse que el servicio de directorio es importante en sí mismo, pero es fundamental para aglutinar información que puede ser fuente de objeto para desplegar nuevos servicios basados en la cooperación entre las distintas aplicaciones y el servicio de directorio.

Así, el servicio de directorio puede actuar como servidor de autenticación, proporcionando el servicio de contraseña única. Además puede contener información necesaria para que los distintos servidores puedan decidir si un usuario puede acceder a determinada información.

Se puede utilizar el servicio de directorio como repositorio en el cual almacenar la información que varios servidores deben compartir, por ejemplo: la configuración, información sobre el control de acceso, etc.

Además, el directorio proporciona un protocolo estándar para gestionar toda la información contenida en él evitando la necesidad de desarrollar dicho protocolo.

Otra utilidad que puede resultar interesante es la de emplear el servicio de directorio para indexar la documentación almacenada en el servidor Web, con la precisión que otras herramientas no pueden generar.

Debido a XML, los documentos contarán con metainformación, es decir, información sobre la información que contienen, lo cual hará más fácil y eficaz la labor de indexación de los contenidos del servidor Web. Aquí es donde el servicio de directorio puede jugar un papel importante, ya que proporciona un acceso uniforme a la información contenida en él.

Esta última puede ser una de las mayores utilidades de los directorios, ya que permiten separar la operación de localización de la información del servidor que la contiene.

2.5. El formato de intercambio de datos LDIF.

El *LDAP Data Interchange Format* (LDIF) es un formato que se utiliza para la importación y exportación de datos independientemente del servidor LDAP que se esté utilizando.

Cada servidor LDAP tiene una o varias maneras de almacenar físicamente sus datos en el disco, por esto que LDIF provee una forma de unificar la manera de tratar los datos y así poder migrar de un servidor a otro sin importar que clase de implementación es.

El formato LDIF es simplemente un formato de texto ASCII para entradas LDAP, que tiene la siguiente forma:

```
dn: <nombre distinguido>
<nombre_atributo>: <valor>
<nombre_atributo>: <valor>
<nombre_atributo>: <valor>
```

Entonces, una entrada del directorio en formato de intercambio de datos LDIF consiste en dos partes:

- El DN que debe figurar en la primera línea de la entrada y que se compone de la cadena **dn**: seguida del **nombre distinguido** (DN) de la entrada.
- La segunda parte son los atributos de la entrada. Cada atributo se compone de un nombre de atributo, seguido del carácter dos puntos ':' y el valor del atributo. Si hay atributos multivaluados deben ponerse seguidos.

  No existe ningún orden preestablecido para la colocación de los atributos, pero es conveniente listar primero el atributo **objectclass**, para mejorar la legibilidad de la entrada.

En un archivo LDIF puede haber más de una entrada definida, cada entrada se separa de las demás por una línea en blanco. A su vez, cada entrada puede tener una cantidad arbitraria de pares `<nombre_atributo>: <valor>`.

Este formato es útil tanto para realizar copias de seguridad de los datos de un servidor LDAP, como para importar pequeños cambios que se necesiten realizar manualmente en los datos, siempre manteniendo la independencia de la implementación LDAP y de la plataforma donde esté instalada.

A continuación, se puede observar un ejemplo de una entrada para describir una cuenta de usuario en un servidor:

```
dn: uid=upruebas,ou=People,dc=ejemplo,dc=com
objectclass: account
objectclass: posixAccount
objectclass: topuid: upruebas
cn: Usuario Pruebas
loginshell: /bin/bash
uidnumber: 512
gidnumber:  300
homedirectory: /home/upruebas
gecos: Usuario Pruebas
```

```
userpassword: 123456
```

3. VIRTUALIZACIÓN.

A la hora de desarrollar una aplicación web, se debe tener en cuenta que el entorno de producción no es el mismo que el de desarrollo. Estas diferencias suelen provocar errores, al desplegar la aplicación, que no se producen durante el desarrollo local. Para resolver estos problemas se suele recurrir a dos técnicas diferentes, **y que pueden utilizarse combinadas:** la utilización de **contenedores y la virtualización**.

Los contenedores permiten configurar los entornos de desarrollo de manera que pueden reproducirse de forma idéntica en cualquier máquina, independientemente del sistema operativo y la configuración del anfitrión. De este modo, se puede utilizar la misma configuración para el servidor de despliegue, el servidor de pruebas y los equipos locales de desarrollo.

La virtualización consiste en crear una máquina virtual con una configuración similar a la del servidor de desarrollo. Para facilitar esta tarea, hay programas como **Vagrant**, que permiten configurar su entorno de desarrollo de la misma manera a un mismo equipo de programadores, ya que **permite crear las máquinas virtuales y aprovisionarse a partir de un mismo archivo de configuración** en lugar de hacer las instalaciones individualmente.

3.1. Contendores

Hay que tener en cuenta que a la hora de desarrollar y desplegar una aplicación web es habitual tener que trabajar en diferentes entornos (configuraciones de hardware y software). Por un lado, está el entorno de desarrollo, que es donde se desarrolla la aplicación (probablemente el ordenador personal) y, por otro, está el entorno de pruebas donde se comprueba que la aplicación funciona correctamente antes de hacer el despliegue (este entorno puede coincidir con el de desarrollo o encontrarse en otra máquina en una red local o remota). Finalmente, está el entorno de producción, que se encuentra en un servidor remoto.

Como cada uno de estos entornos puede tener una configuración de hardware y software diferente, a menudo hay problemas en desplegar aplicaciones web: mientras que en el entorno de desarrollo puede funcionar perfectamente, en desplegarla a cualquiera de los otros entornos pueden producirse errores provocados por las diferencias entre sistemas operativos, servidores web, bibliotecas instaladas o, incluso, por el sistema de ficheros.

Para minimizar estos problemas se puede recurrir a la utilización de sistemas de contenedores (y así nos aseguramos de que la configuración del entorno de pruebas y el de desarrollo es idéntica) y la virtualización (para crear entornos de desarrollos lo más similar posible a la entorno de producción o desarrollo).

Aunque la utilización de contenedores y máquinas virtuales puede parecer muy similar, no se trata del mismo concepto.

- **Los contenedores** son mucho más ligeros y prácticamente no afectan el rendimiento de la aplicación, lo que posibilita su utilización en el desarrollo.
- **Las máquinas virtuales** son instalaciones completas del sistema operativo y añaden capas extras a la ejecución de los programas, ya que cada instrucción debe pasar por el sistema operativo virtualizado, el software de virtualización, el hardware de virtualización del equipo huésped y el núcleo del sistema operativo huésped.

Los contenedores utilizan el mismo núcleo del sistema operativo y, por este motivo, no hay ninguna disminución apreciable de rendimiento. Además, el espacio que ocupa en disco es muy reducido, ya que sólo hay que añadir los archivos específicos que requiere la aplicación que se ejecuta en el contenedor.

En el caso de sistemas distribuidos (una aplicación desplegada en múltiples máquinas), es muy útil utilizar contenedores, ya que sólo hay que preparar el contenedor una vez e instalarlo en tantos servidores como sea necesario. Este es uno de los motivos por los que Google utiliza contenedores para desplegar sus aplicaciones en lugar de tener que configurar miles de equipos individualmente.

Docker es un sistema de contenedores basado en el núcleo de Linux. Se trata de software libre y entre los principales colaboradores hay empresas como Google, IBM, Cisco, Microsoft y Red Hat.

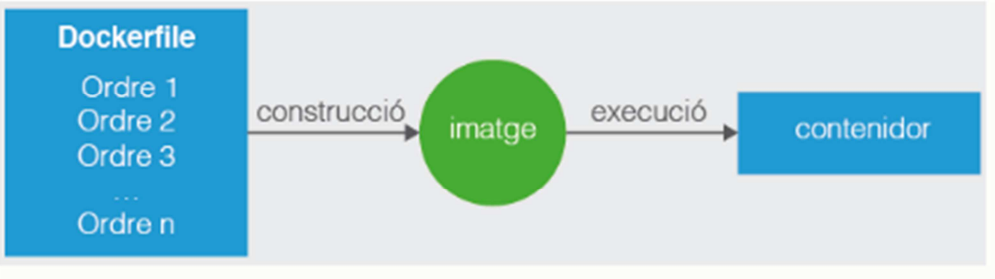
Como se trata de un contenedor basado en Linux, no puede utilizarse directamente a máquinas con otros sistemas operativos, pero es posible utilizarlo en otros **entornos de desarrollo mediante la virtualización**. En su web hay enlaces a **instaladores** que simplifican esta tarea e incluyen todos los archivos necesarios para instalar **Docker Linux, Mac y Windows.**

Algunas ventajas de desplegar una aplicación utilizando Docker son los siguientes:

- La aplicación funciona igual en el servidor de pruebas que en el de producción porque el entorno es el mismo.
- Cada aplicación se encuentra aislada del resto de aplicaciones, en su propio contenedor.
- No es necesario instalar ningún otro componente, además de Docker, para ejecutar la aplicación en otro equipo. La aplicación funciona directamente en instalar el contenedor, para que todas las dependencias se encuentran en el contenedor.

Los contenedores de **Docker son instancias de una imagen** que contiene todos los archivos necesarios empaquetados para crear el contenedor en un solo archivo. A su vez, esta imagen es construida a partir de un archivo llamado

Dockerfile, que contiene todas las órdenes necesarias para construir dicha imagen.



Podéis encontrar más información sobre Docker y ejemplos en:

https://ioc.xtec.cat/materials/FP/Materials/ICC0_DAW/DAW_ICC0_M08/web/html/WebContent/u3/a1/continguts.html
https://guiadev.com/introduccion-a-docker/
https://goo.gl/N44YOg

3.2. Virtualización

En el caso de las **máquinas virtuales**, cada máquina se encuentra completamente aislada de la máquina huésped y, por consiguiente, esto afecta al rendimiento: cada acción debe ser procesada por la máquina virtual, el software de virtualización y la máquina huésped. Además, la **instalación** del software de cada máquina **debe ser completa**, es decir, debe incluir como mínimo todos los archivos del sistema operativo para funcionar y, por tanto, el espacio en disco necesario será mucho más mayor que en el caso de los contenedores.

Por otra parte, como se trata de una instalación completa, es posible utilizar máquinas virtuales con diferentes sistemas operativos independientemente de cuál esté instalado en la máquina huésped (por ejemplo, una máquina virtual con Linux en una máquina huésped con Windows).

Cabe destacar VirtualBox y VMware como software de virtualización porque son los más populares y porque ofrecen versiones gratuitas. Ambos pueden utilizarse para realizar instalaciones completas de entornos de desarrollo. Eso sí: en lugar de acceder directamente a la máquina creada, se suele utilizar otros programas como **Vagrant**, que gestionan el software de virtualización para generar estos entornos, **automatizando parte de las tareas y simplificando muchas acciones habituales, como puede ser la creación de una máquina base a partir de un repositorio, el aprovisionamiento de la máquina o la configuración de las carpetas compartidas**.

Vagrant (www.vagrantup.com) es una tecnología basada en la virtualización que permite crear entornos de desarrollo portables y fácilmente reproducibles. Gracias a la virtualización es posible trabajar con diferentes versiones de sistemas operativos en la misma máquina.

Hay que tener en cuenta que a la hora de trabajar en diferentes proyectos, las características de las máquinas de producción muy raramente serán las mismas que las del equipo propio de desarrollo. Es decir, ni el sistema operativo, ni las versiones de los servidores de bases de datos, ni las versiones de los servidores web se corresponderán.

Para entender cuál es el problema que soluciona Vagrant véase el ejemplo siguiente:

Trabajar en múltiples proyectos localmente

> *Un profesional trabaja en dos proyectos al mismo tiempo como desarrollador, y la máquina de desarrollo utiliza Windows 10.*
>
> *El proyecto actual es una aplicación en PHP 7, desplegada en un servidor Red Hat Enterprise Linux 7.3, que utiliza Nginx como servidor web.*
>
> *Se ha detectado un problema en un proyecto antiguo que debe solucionarse. Este está desarrollado en PHP 5.3, desplegado en un servidor con Ubuntu 2.12, que utiliza Apache como servidor web.*
>
> *Para poder trabajar con ambos proyectos a la vez en la misma máquina hay que hacer las siguientes acciones:*
>
> - *Instalar dos servidores web diferentes en su máquina.*
> - *Instalar dos versiones diferentes de PHP.*
> - *Modificar las configuraciones para evitar conflictos. (En producción no se debe hacer, si este cambio se propaga a los servidores, puede provocar errores.)*
> - *Hay que tener en cuenta que cada uno de estos servicios consume recursos, se utilicen o no, a la máquina principal.*
>
> *Cabe destacar otro inconveniente: no se pueden descubrir errores debidos al sistema operativo hasta que no se realice el desarrollo, ya que el de la máquina no se corresponde con los de los proyectos.*

Como se puede apreciar en el ejemplo, en el mejor de los casos han tenido que instalar múltiples servicios en el ordenador y no hay ninguna garantía de que cuando se despliegue, la aplicación funcione correctamente. Además, si en el equipo hay más de un desarrollador trabajando en los mismos proyectos, se deben repetir estos pasos para cada uno de los entornos de cada desarrollador implicado.

Piense que las configuraciones de estos otros equipos también pueden ser diferentes unas de otras, y posiblemente no han trabajado en los mismos proyectos. Por consiguiente, cuando se instalen los nuevos servicios, se pueden presentar nuevos problemas y conflictos.

Otra opción es **trabajar directamente sobre servidores remotos**. En este caso, para cada proyecto se consigue una configuración muy cercana a la de desarrollo, pero presenta los inconvenientes siguientes:

- Se incrementa el coste de desarrollo, ya que se tienen que pagar servidores extras para cada proyecto.
- Múltiples desarrolladores no pueden trabajar sobre el mismo servidor directamente, ya que se deben transferir los cambios y se podrían sobrescribir los archivos.
- Según la configuración de estos servidores y otras herramientas de desarrollo, muchas tareas deben realizarse utilizando la línea de comandos y editores de texto que no admiten el uso del ratón.
- En caso de que hubiera problemas de conectividad, no se podría continuar trabajando.

En cambio, **si se utilizan máquinas virtuales, se puede trabajar con tantas configuraciones diferentes como sea necesario, independientemente del sistema operativo original.** Este sistema soluciona prácticamente todos los problemas detectados en los métodos anteriores:

- Las configuraciones están encapsuladas, no hay conflictos entre una y otra.
- La configuración es muy cercana a la de la máquina de desarrollo.
- Cada desarrollador trabaja en su máquina, no hay peligro de perder datos.
- Se pueden utilizar las herramientas de desarrollo preferidas por el programador, ya que se pueden pasar los datos entre máquinas utilizando carpetas compartidas.
- No hay ningún coste añadido para que no se han de contratar nuevos servidores.
- No se pueden producir problemas de conectividad, ya que todo se encuentra en la misma máquina.

Aunque es una muy buena opción, presenta el inconveniente de tener que crear múltiples máquinas virtuales cuando se trabaja en un equipo de desarrolladores, ya que cada ordenador se debe crear la máquina virtual para el proyecto y hacer todo lo proceso de instalación del sistema operativo, los servicios y la configuración o copiar los archivos de una de las máquinas del otro desarrollador al nuevo.

Afortunadamente, **Vagrant soluciona este problema usando un archivo de configuración que permite, rápidamente, recrear cualquier máquina sin interacción del usuario, aunque una vez están instaladas**, los desarrolladores pueden interactuar utilizando la línea de comandos o directamente, mediante una carpeta compartida.

Además, cuando no se necesitan, se pueden destruir para recuperar el espacio en el disco, ya que se necesita muy poco tiempo para recrearlas y no es nada complejo. En este caso, sólo se pierden los datos creadas dentro de la máquina: las de la carpeta compartida no se pierden, que es la carpeta donde se recomienda trabajar.

Otra ventaja es que Vagrant permite generar múltiples máquinas a partir de un solo archivo de configuración, por lo que se pueden simular interacciones complejas que no son posibles cuando se trabaja con un único ordenador.

En resumen, la utilización de **Vagrant presenta las siguientes ventajas**:

- El entorno de desarrollo es similar al de producción.
- Se pueden mantener múltiples entornos de desarrollo en la misma máquina.
- Se puede desplegar el entorno de desarrollo en cuestión de minutos.
- Todos los miembros del equipo de desarrollo trabajan con el mismo entorno.
- Se trabaja con el código mediante carpetas compartidas.
- Cuando un proyecto no es necesario, se puede destruir la máquina virtual, porque para reconstruirla (si fuera necesario en el futuro) sólo se necesita el archivo Vagranfile.
- Se pueden simular conjuntos de máquinas.

Podéis encontrar más información sobre Docker y ejemplos en:

https://ioc.xtec.cat/materials/FP/Materials/ICC0_DAW/DAW_ICC0_M08/web/html/WebContent/u3/a1/continguts.html
https://www.vagrantup.com/
https://guiadev.com/docker-vs-vagrant/

La sección 3.Virtualización es una adaptación del original proporcionado por el IOC (Institut Obert de Catalunya):
https://ioc.xtec.cat/materials/FP/Materials/ICC0_DAW/DAW_ICC0_M08/web/html/WebContent/u2/a1/continguts.html