

# **Introducción a la World Wide Web**

# Tabla de Contenidos

Introducción a la World Wide Web.....	1
Tabla de Contenidos.....	2
Introducción a la Wold Wide Web .....	3
¿Que se entiende por WWW ? .....	3
Un Poco de Historia.....	3
Internet, Red de Redes.....	4
Servicio DNS (Domain Name Service) .....	8
Protocolo HTTP (HyperText Transfer Protocol).....	11
Comandos HTTP .....	16
Códigos de Respuesta HTTP .....	16
Referencias.....	20

# Introducción a la Wold Wide Web

## ¿Que se entiende por WWW ?

Sintéticamente podemos decir que el WWW (World Wide Web) es un sistema de información, soportado por redes de ordenadores, especialmente aquellas basadas en el protocolo TCP/IP, y basado en una arquitectura cliente/servidor. Se deben reunir tres elementos para que opere: un cliente, un servidor y una plataforma de red que los conecte.

Asimismo se puede definir el WWW como un servicio de visualización de documentos basados en un lenguaje especial de marcas, el HTML, en un protocolo especial para la transferencia de los mismos, el HTTP, y en un concepto de enlaces entre diferentes documentos, el HIPERTEXTO. Sin duda alguna, este servicio es el más conocido de INTERNET y el de más repercusión social.

Como vemos para el funcionamiento correcto del servicio Web deben existir una serie de elementos combinados que en conjunto le dan a la Web su esencia: El HTML (Hypertext Markup Language) que proporciona la codificación necesaria, un software cliente (*browser*) quien realiza la decodificación en el usuario final, un software especializado que atienda las peticiones de los usuarios y les de curso (*HTTP Server*), y la propia red basada en TCP/IP y su protocolo especializado (el *HTTP, Hypertext Transfer Protocol*) encargados de realizar el transporte entre los proveedores y los usuarios finales.

Por lo tanto antes de comenzar a escribir nuestras propias páginas Web, debemos conocer un poco más cómo funcionan cada uno de estos elementos.

## Un Poco de Historia

El concepto de hipertexto fue acuñado por Ted Nelson, en 1965, y se basa en la idea general de utilizar marcas de texto especiales dentro de un documento que permiten, al ser activados, enlazar con otra parte del documento o con otro documento diferente.

Partiendo de esta idea y basándose en los servicios preexistentes, en 1989 un joven investigador del CERN, Tim Berners-Lee, tuvo la idea de crear un entorno para compartir información, de manera que un documento fuese accesible por cualquier ordenador conectado a INTERNET, sin importar dónde estuvieran el ordenador que tuviera el documento o el ordenador que lo solicitaba. Además, debería ser posible enlazar mediante hipertexto diferentes documentos o recursos sin límites de localización geográfica, de tal forma que el usuario pudiera “navegar” de forma transparente por dichos documentos.

El problema principal que debía resolver el equipo de desarrollo de esta tecnología era como intercambiar volúmenes importantes de información (una de las características que se buscaba era la multimedia) sobre una red de una amplia distribución geográfica y de concepción heterogénea.

Para solucionar estos ítems se crearon nuevas especificaciones, nuevas formas de comunicar equipos informáticos y nuevas implementaciones de carácter general.

Así fue que nacieron:

El protocolo HTTP (HyperText Transfer Protocol), que permitía la transferencia de documentos basados en texto (en formato ASCII) más ciertas cabeceras que aportaban el control de transmisión, en forma rápida y segura.

Se creó un método de identificación único para un recurso determinado en Internet, denominado URL (Uniform Resource Locator); este identificador indica tanto la localización exacta del recurso como el protocolo necesario para su transferencia.

Ejemplo:

```
http://servidor.dominio/carpeta/pagina.html
```

```
ftp://200.32.56.19
```

Para poder transferir documentos con formato, multimedia y dar soporte hipertextual requiriendo el menor ancho de banda posible se creó un lenguaje de codificación específico, el HTML (HyperText Mark-up Language – Lenguaje de Marcas de Hipertexto), que permite asignar un formato especial de presentación a los elementos del documento contenidos entre unas etiquetas especiales, denominadas marcas o tags.

Por último, del lado del cliente del servicio se creó una herramienta de software específica para poder interpretar y visualizar correctamente los documentos Web, los llamados navegadores (browsers).

Estas fueron los pilares esenciales de los cuales se construyó la WWW (World Wide Web), la gran “telaraña” mundial de páginas Web (documentos Web visualizados en un navegador). El WWW es un sistema de información global, público e independiente, mediante el cual un usuario cualquiera puede acceder a documentos HTML almacenados en diferentes servidores ubicados en cualquier parte del mundo, pudiendo saltar de un servidor a otro de forma instantánea mediante los enlaces de hipertexto contenidos en las páginas Web.

## **Internet, Red de Redes**

Como mencionamos en la introducción de este documento, para poder entender el Web debemos ante todo entender el medio en que se desarrolló, esto es la Internet.

Como todos seguramente sabemos, Internet es una red de computadoras, esto es, una serie de computadoras conectadas entre sí por algún medio físico (Ca-

bles, Fibras Ópticas, Radio enlaces, Satélites, etc.) con el fin de compartir recursos físicos y lógicos entre ellas.

Dentro de las redes de computadoras, Internet es una red de las denominadas WAN (Wide Area Network), o sea una red de área amplia, que se puede interpretar como aquella red en la cuál las conexiones entre los diferentes nodos están fuera de nuestra área de control, de manera que para enviar información entre dos nodos de dicha red, los paquetes de dicha conexión atravesarán redes y/o medios de conexión sobre los cuáles nosotros como emisores y/o receptores no tenemos el más absoluto control.

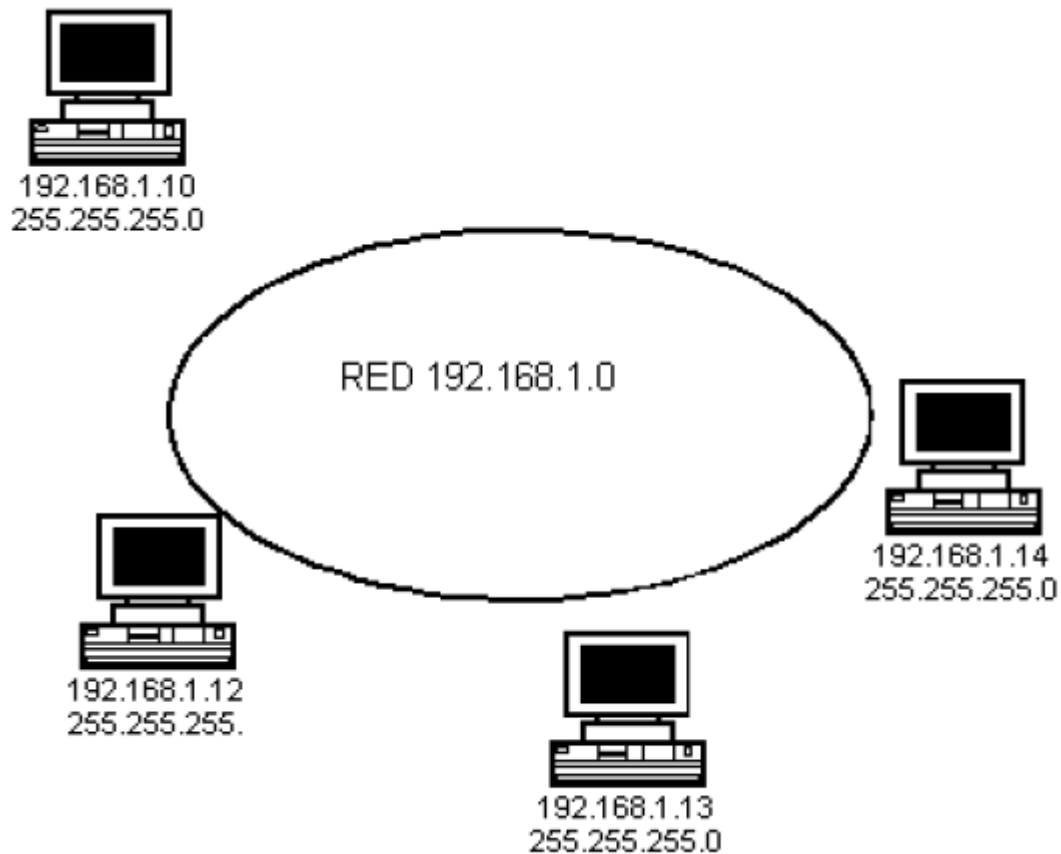
Este último punto es muy importante puesto que en gran medida es lo que dio forma a los protocolos y métodos de comunicación de datos sobre Internet.

Por otra parte Internet es lo que se llama una red de conmutación de paquetes, basada en el conjunto de protocolos conocidos bajo el genérico de TCP/IP. El funcionamiento de una red conmutada de paquetes tiene como una de sus características principales el que parte de la “inteligencia” de la misma, o sea la capacidad de lograr que un paquete determinado llegue a su destino, reside en la propia red además de en sus nodos, lo que produce una red altamente flexible. En el caso específico de una red Wan basada en TCP/IP como Internet, la “inteligencia” está basada en los elementos conocidos como “routers” o encaminadores, quienes tienen a su cargo el encaminamiento de paquetes a lo largo de una red de topología compleja y cambiante como Internet.

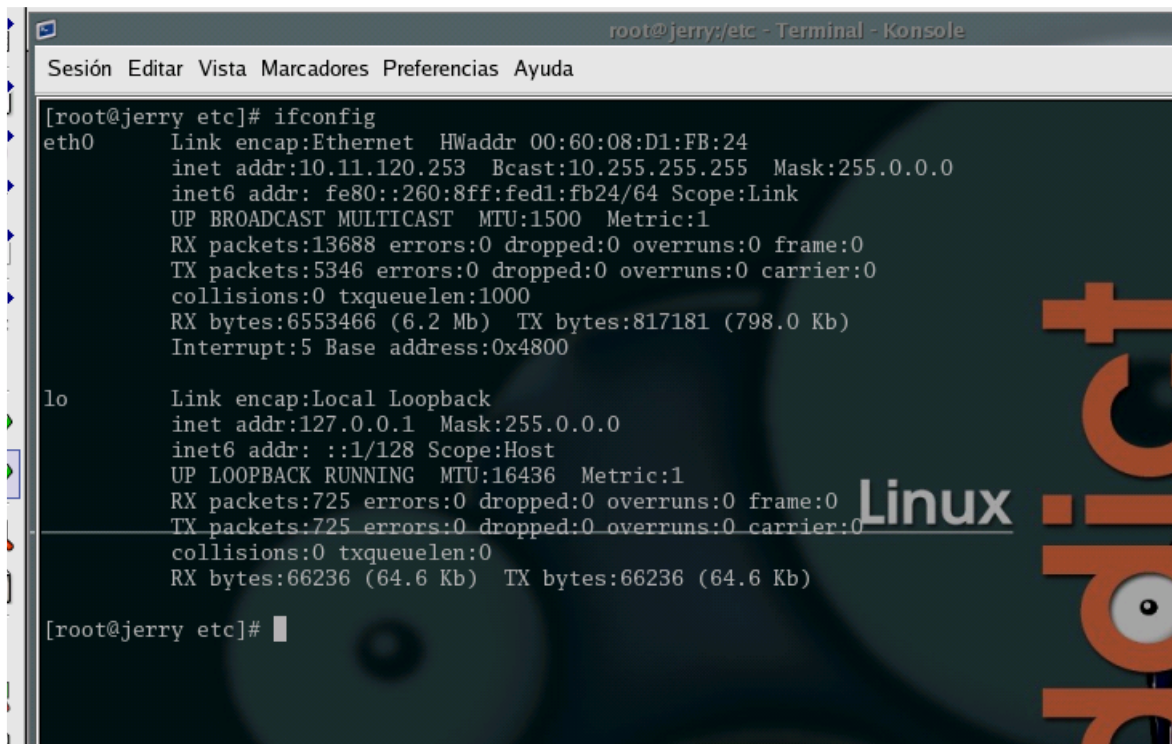
Para comprender más acabadamente como funciona una red basada en TCP/IP, primero debemos tener en claro algunos conceptos. En una red TCP/IP, todo nodo tiene una identificación asociada, dicha asociación es única a lo largo de esta red y se conoce como Dirección IP; haciendo una analogía podemos decir que esta Dirección IP es el “número de teléfono” de un nodo determinado; por lo tanto para “llamar”, comunicarnos con otro nodo a fin de acceder a algún servicio que el brinde, debemos ante todo conocer dicho número de teléfono (Dirección IP), sobre la forma de conocer esta Dirección IP volveremos más adelante cuando hablemos del servicio DNS.

Esta analogía no es gratuita, puesto que el comportamiento de una red WAN basada en TCP/IP guarda bastante parecido con el comportamiento de una red telefónica pública. La dirección IP (en su versión actual IPV4) está compuesta por dos grupos de números de 4 bytes cada uno, que representan respectivamente la dirección y la máscara de red. Por ejemplo una dirección IP válida es 200.56.78.99 y su máscara podría ser 255.255.255.0. Siguiendo con la analogía planteada la máscara de red es la que determina que parte de la dirección IP es la “característica” y que parte el “número de abonado”, esto es en el caso anterior, los tres primeros bytes de la dirección ip (200.56.78) representan la identificación de la red a la que pertenece el nodo (“característica”), y el último byte (99) es específicamente el nodo al que estamos haciendo referencia dentro de esa red; de esta manera todos los nodos que se hallen en la red local de dicho nodo, tendrán los mismos 3

primeros bytes y se diferenciarán por el último byte. Por lo tanto una red con máscara de 24 bits sólo permite tener 254 nodos conectados (Fig. 1).



El comportamiento de un nodo que quiere comunicarse con otro varía si el destinatario está en el mismo segmento IP que él o en otro, esto definido por supuesto por la máscara de red (Fig 2).



```
root@jerry/etc - Terminal - Konsole
Sesión Editar Vista Marcadores Preferencias Ayuda

[root@jerry etc]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:60:08:D1:FB:24
          inet addr:10.11.120.253  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::260:8ff:fed1:fb24/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:13688 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5346 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6553466 (6.2 Mb)  TX bytes:817181 (798.0 Kb)
          Interrupt:5 Base address:0x4800

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:725 errors:0 dropped:0 overruns:0 frame:0
          TX packets:725 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:66236 (64.6 Kb)  TX bytes:66236 (64.6 Kb)

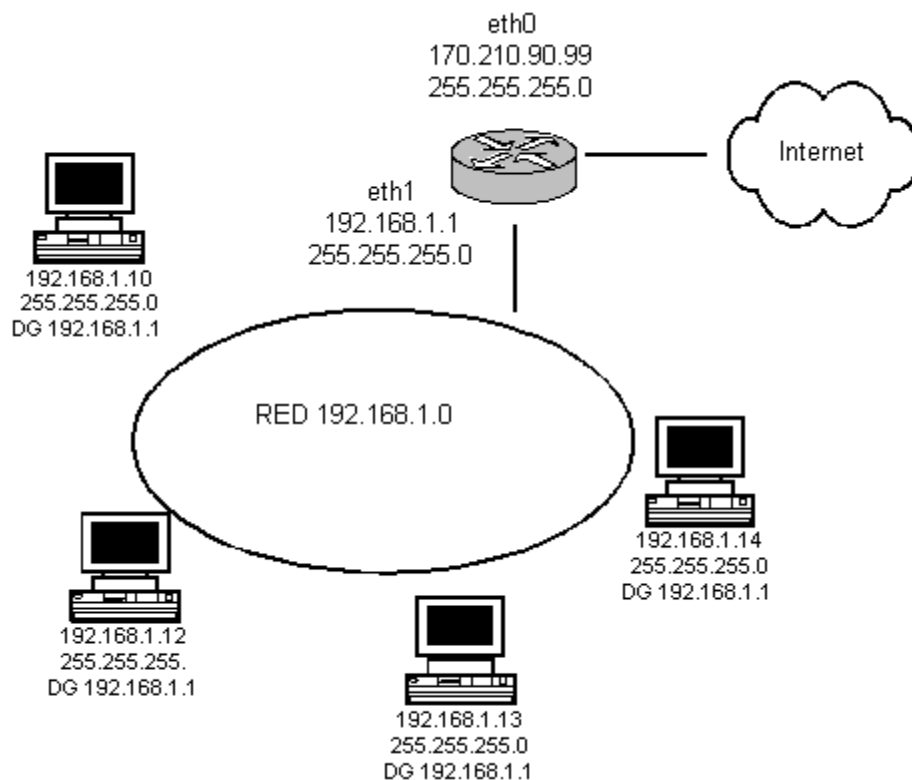
[root@jerry etc]#
```

Si está en el mismo segmento IP, el nodo simplemente obtiene la dirección MAC (Dirección física de la interfaz de red) del destinatario a través de un protocolo llamado ARP (Address resolution protocol) que básicamente trabaja a base de difusión, y una vez que obtiene la dirección MAC, utiliza la capa de enlace para comunicarse.

El ARP (**A**ddress **R**esolution **P**rotocol) es un protocolo de la familia TCP/IP usado para obtener la dirección física de un nodo (Dirección MAC). Un nodo cliente difunde (broadcast) una petición ARP sobre la red con la Dirección IP del nodo con el cual se quiere comunicar y el nodo con dicha Dirección IP responde enviando su dirección física, así los paquetes TCP/IP pueden ser intercambiados. Resumiendo el protocolo ARP devuelve una dirección de capa 2 para una dirección de capa 3. Si el destinatario no está en su mismo segmento IP, el nodo no tiene forma de obtener su MAC, y por lo tanto no se puede comunicar; aquí es donde entran en juego los encaminadores. Cada nodo además del par Dirección IP/Máscara tiene como mínimo otra Dirección IP asociada denominada puerta de enlace o *Default Gateway*, esta dirección es la de un nodo alcanzable por ARP dentro de la red donde se encuentra el nodo original y que recibirá todos los paquetes que envíe el nodo cuyo destino no pueda ser alcanzado por ARP (no estén en su misma subred, determinada esta por la máscara de red).

Este nodo denominado Puerta de enlace es el famoso encaminador o *router*, puesto que es el que se encarga de encaminar los paquetes desde una red hacia otra. Un *router* puede ser desde una computadora personal corriendo un software específico hasta un equipo específicamente diseñado para este fin.

Una vez que el paquete lo tiene el *router*, este toma una decisión de encaminamiento basado en los algoritmos que tenga instalados, pero básicamente observa si está conectado físicamente a la red de destino, por supuesto consultando sus pares Dirección IP/Máscara (cabe aclarar que en todos los casos, los *routers*, son lo que se denomina *host multibase*, nodos con más de un enlace físico), si es así, simplemente envía el paquete a esa red. Si no está conectado físicamente a la red de destino lo envía a otro *router*, su propio Default Gateway, o toma una decisión de encaminamiento en caso de tener mas de un *Gateway* definido. Esta decisión de encaminamiento dependerá de lo complejo que sea el encaminador y del algoritmo o algoritmos que utilice. Estos algoritmos pueden ser muy simples, como enviarlo directamente a una Puerta de Enlace del propio *router* (ruta estática) o muy complejos, compilando estadísticas de tráfico, de disponibilidad de enlace, etc. y tomando decisiones en base a ello (Fig. 3).



## Servicio DNS (Domain Name Service)

Hasta aquí hemos explicado sucintamente el funcionamiento de una red IP, pero la mayoría de Uds. nunca, navegando en Internet, se han topado con una Dirección IP, si no con nombres de recursos del tipo [www.loqueequieras.com.ar](http://www.loqueequieras.com.ar), entonces, ¿Dónde se realiza el cambio entre Dirección IP y nombre de recurso?. En un principio en Internet solo existían Direcciones IP numéricas, tal como explicamos



en el punto anterior, y un servicio de nombres muy limitado basado en el llamado archivo *hosts*. Este es un archivo que se distribuye en casi todos los sistemas operativos y consiste en una lista estática del tipo Dirección IP / Nombre de Host.

- En Unix/Linux: En general se encuentra en el directorio */etc*.
- En Windows 9x/me: En general se encuentra en el directorio *c:\windows*
- En Windows 2000: En general se encuentra en el directorio *c:\winnt\system32\drivers\etc*
- En Windows XP: En general se encuentra en el directorio *c:\windows\system32\drivers\etc*

En la figura 4 vemos un ejemplo del archivo *hosts*, en este caso en Linux.

A screenshot of a Linux terminal window titled "root@jerry:/etc - Terminal - Konsole". The menu bar includes "Sesión", "Editar", "Vista", "Marcadores", "Preferencias", and "Ayuda". The terminal displays the contents of the /etc/hosts file:

```
# Do not remove the following line, or various programs  
# that require network functionality will fail.  
127.0.0.1    jerry    localhost.localdomain  localhost  
192.168.7.9  gizmo    gizmo.fff.com.ar
```

The background features a dark blue gradient with faint circular patterns. On the right side, there is a large, stylized orange logo that reads "tictac" vertically, with a cartoon character's face partially visible behind it. At the bottom left, a status bar indicates: "hosts" 4 lines, 187 characters.

El método de emparejar Direcciones IP con nombres de recursos mediante el archivo *hosts*, funcional en un principio, empezó a quedar chico a medida que la cantidad de nodos en Internet crecía; esto se debía básicamente a tres elementos:

- Los archivos *hosts* se almacenan en cada nodo individualmente, por lo que para actualizar la lista de nodos, hay que actualizar todos los archivos *hosts* de la red.
- Los archivos *hosts* son planos, esto es no permiten armar una estructura jerárquica de nombres.
- Los archivos *hosts* se deben actualizar a mano, no existe un servicio que se encargue de hacerlo automáticamente.

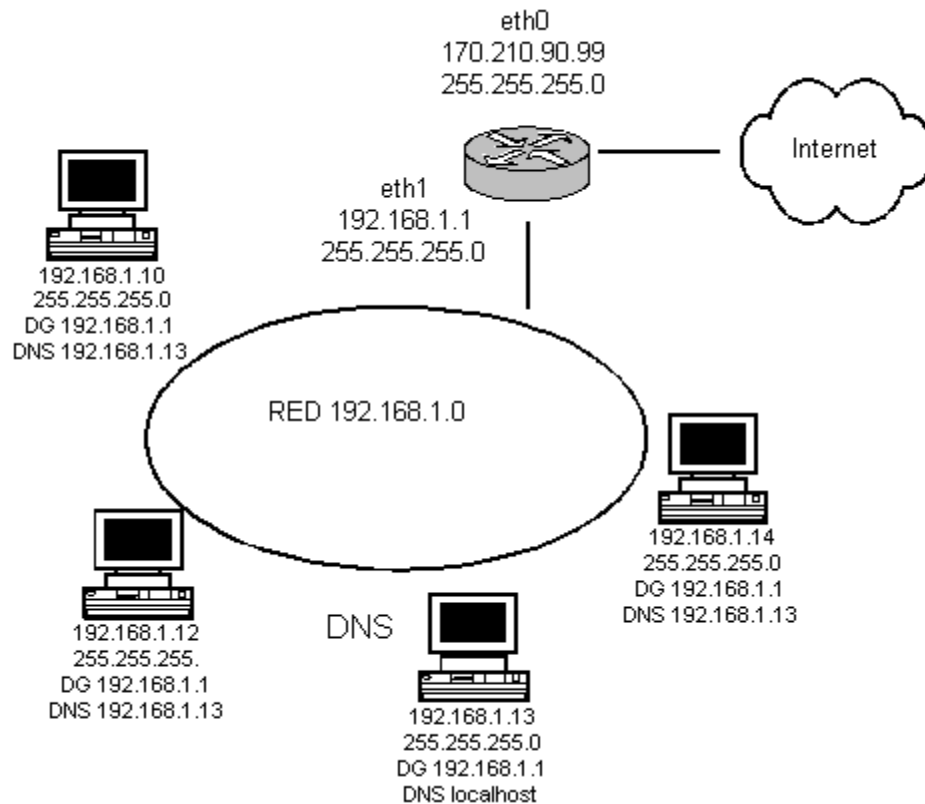
En base a estos problemas, se creó un servicio (software) de red que permitiera salvar dichos problemas y otorgara la funcionalidad de la traducción entre nombre de recurso (nombre de dominio) y Dirección IP; este servicio se denomina DNS (Domain Name System).

El servicio DNS es básicamente una base de datos jerárquica y distribuida a lo largo de la red, que almacena tablas de doble entrada en el formato Nombre de Dominio/Dirección IP.

Estos nombres de dominio tienen una estructura definida formada por diferentes niveles, a saber tomando una dirección como [www.infobae.com.ar](http://www.infobae.com.ar), la dirección de dominio se lee de derecha a izquierda y se interpreta como:

- .ar = Indicativo de país, en este caso indica que el dominio está registrado en la entidad de administración de nombres de la Argentina (nic.ar)
- .com = Indicativo de tipo de nodo, .com indica un nodo con fines comerciales.
- .infobae = Es el nombre de la red a la que nos dirigimos (en realidad infobae.com.ar), en general un nombre significativo respecto del servicio que brinda el nodo.
- www. = El nombre del nodo específico al que queremos contactar, en este caso el nombre además indicaría que este es un servidor de Web, pero no es necesariamente así.

Por lo tanto un nodo determinado conectado a Internet, si quiere acceder a recursos por medio de su nombre de dominio, deberá tener especificada la Dirección IP de al menos un servidor de DNS al que debe poder acceder en forma local o por medio de encaminamiento (Fig. 5).



## Protocolo HTTP (HyperText Transfer Protocol)

Como explicamos en la introducción, cualquiera que quiera entender la Web, debe por lo menos tener idea de sus tres bases fundamentales, la red, el protocolo de transferencia y el lenguaje de codificación.

El protocolo HTTP (HyperText Transfer Protocol) es el protocolo de transferencia principal en el funcionamiento de la Web, es básicamente un protocolo de transferencia de texto, muy simple en su funcionamiento, y muy flexible. Este protocolo es parte del llamado conjunto de protocolos TCP/IP, es un protocolo correspondiente a la capa de aplicación del modelo de la OSI, orientado a objetos, genérico y stateless (no mantiene el estado de la conexión), y en general asociado al puerto TCP 80.

En síntesis podemos decir que las propiedades generales del protocolo HTTP son:

- Un esquema de direccionamiento específico: El HTTP usa el concepto de referencia provisto por el URI (Uniform Resource Indicator); el URL (Uniform Resource Location) y el URN (Uniform Resource Name). Para indicar un recurso a través de este método se debe especificar de la forma servicio://host/archivo.extensión, y de esta forma HTTP puede lidiar con los servicios más básicos de Internet.

- Arquitectura Cliente/Servidor: El HTTP está basado en el paradigma petición/respuesta, donde un cliente pone peticiones que son respondidas por un servidor, en general sobre una conexión TCP al puerto 80.
- El protocolo es HTTP es *connectionless* y *stateless*: Luego que el servidor respondió a una petición de un cliente, toda la información acerca de la transacción es eliminada, no existe memoria entre peticiones de clientes, en una implementación pura de HTTP, cada petición es tomada como una petición nueva.
- Existe una representación de tipos de datos abierta y extensible: HTTP usa los *Internet Media Types* (antes conocidos como MIME Content-type, <http://www.iana.org/assignments/media-types/index.html>) para proveer una manera flexible de intercambiar datos entre el cliente y el servidor de manera que cada uno de ellos sepa como tratar los datos que intercambia.
- El protocolo HTTP se encuentra actualmente en su versión 1.2 (RFC 2774), que mejora sustancialmente al la anterior, el HTTP 1.1, sobre todo en lo que respecta al manejo de conexiones entre navegador cliente y servidor Web.

Básicamente el protocolo HTTP implementa tres comandos que se pueden utilizar para comunicarse con el servidor Web, a saber: GET, POST y HEAD.

El método GET es el más antiguo de los tres y el único que implementaba la versión original de HTTP (0.9), y consiste en el comando GET seguido de la URI (Uniform Resource Identifier) o el URL (Uniform Resource Locator) o el URN (Uniform resource Name), más dos retornos de carro (adicionalmente se puede especificar la versión del protocolo), esto simplemente le indica al servidor HTTP que inicie la transferencia del documento indicado por la URI/URL/URN.

*Ejemplo:*

```
selena: telnet www.extropia.com 80
```

```
Trying 206.53.239.130...
```

```
Connected to www.extropia.com.
```

```
Escape character is '^]'.
```

```
GET /Scripts/Columns/irobot.html
```

```
<HTML>
```

```
<HEAD>

<TITLE>Hello there</TITLE>

</HEAD>

<BODY>

Hello there. My, you are awfully good-looking

to be a web browser!

</BODY>

</HTML>

Connection closed by foreign host.
```

Adicionalmente el método GET nos permite enviar información hacia el servidor al realizar la petición, la información viaja a través del método conocido como *URL encoded*, esto es como parte de la URL; para esto se especifican la información que se desea enviar como conjuntos de nombre parámetro/valor, a partir del nombre del archivo en el servidor que procesará la petición.

```
GET http://www.clarin.com.ar/buscar.cgi?quebusco=nose
```

En el ejemplo se realiza una petición a la URL [www.clarin.com.ar/buscar.cgi](http://www.clarin.com.ar/buscar.cgi) y se le pasa como parámetro la variable **quebusco** seteada en el valor **nose**; por supuesto el servidor Web en cuestión deberá saber que hacer con estos valores si no simplemente se devolverá el contenido de buscar.cgi.

En el caso que se quiera pasar más de un valor, los parámetros se deberán especificar separados por el símbolo & o por el símbolo +

```
GET http://www.clarin.com.ar/buscar.cgi?quebusco=nose&paraque=sisupiera
```

El método GET sólo admite símbolos alfanuméricos, por lo que cualquier otro valor que se pase deberá estar codificado por su valor ASCII en hexadecimal y antecedido del carácter %, por ejemplo para pasar la variable **paraque=si supiera**, la petición deberá codificarse de la siguiente manera:

```
GET
http://www.clarin.com.ar/buscar.cgi?quebusco=nose&paraque=si%27supiera
```

Ahora, el método GET, pese a ser muy sencillo de utilizar, tiene una serie de deficiencias, a saber:

- El método está limitado a 1024 caracteres de longitud, por lo que si deseamos enviar una petición muy larga, como puede ser una serie de parámetros complejos, o enviar un archivo como parte de la URL, el método fallará.
- Todos los datos que envío como parte de la petición, se envían en la URL, por lo tanto son visibles fácilmente por el usuario lo que implica una vulneración de la seguridad del sitio.

Para evitar esto y otorgar un método más poderoso de intercambiar información con el servidor HTTP, en la implementación 1.0 del protocolo se introdujeron los comandos POST y HEAD, además se cambió fundamentalmente el formato de la transacción, a partir de esta versión, el intercambio de información consiste en encabezados seguidos de una línea en blanco y datos adicionales; por supuesto la implementación del método GET sigue los parámetros anteriores de manera de mantener compatibilidad con servidores mas viejos.

Otro cambio importante en la versión 1.0 es la introducción de los *Internet Media Types* (antes MIME type), en la cabecera de la transacción lo que permitió intercambiar más fácilmente información no alfanumérica informando al navegador y al servidor de HTTP que tipo de información a esperar.

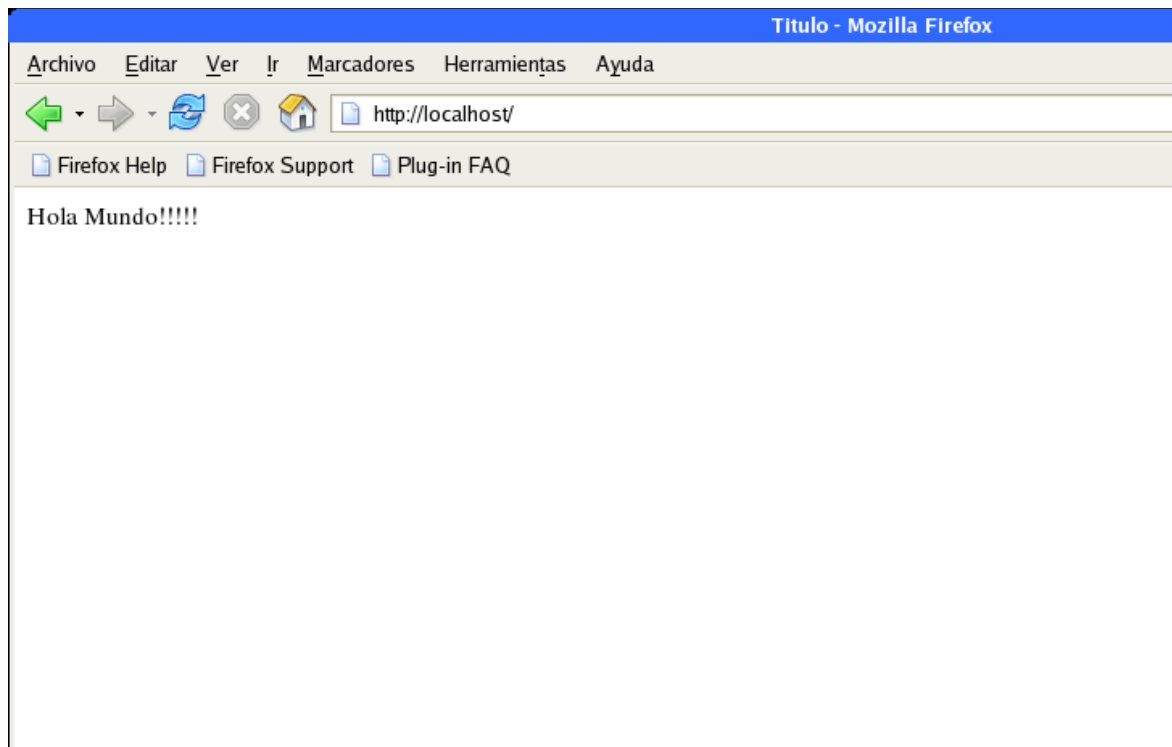
Una transacción tipo GET pero con las cabeceras de la versión 1.1 de HTTP seguirá aproximadamente la siguiente forma (Fig. 6):

```
GET / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-AR; rv:1.7.6) Gecko/20050226 Firefox/1.0.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Mon, 20 Jun 2005 21:17:10 GMT
Server: Apache/2.0.49 (Fedora)
X-Powered-By: PHP/4.3.4
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Titulo</title>
</head>
<body>
Hola Mundo!!!!!!
</body>
</html>
```

En el navegador (Fig. 7):



En cambio una petición de tipo POST esta compuesta del comando POST, una serie de cabeceras que indican entre otras cosas cuál es el tipo de contenido que se va a enviar, y el largo de este contenido, y en el cuerpo de la petición se envía lo que se desea que el servidor procese, ya sean pares de parámetros o por ejemplo un archivo completo para un *upload* (Fig. 8)

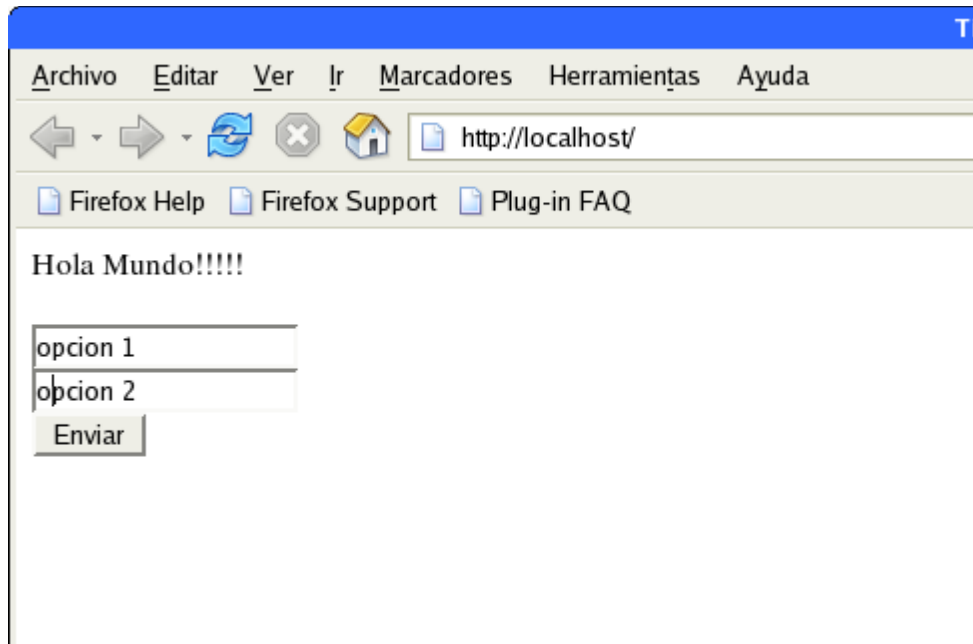
```
POST /index3.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-AR; rv:1.7.6) Gecko/20050226 Firefox/1.0.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://localhost/
Content-Type: application/x-www-form-urlencoded
Content-Length: 45

Campo1=opcion+1&Campo2=opcion+2&Enviar=Enviar

HTTP/1.1 200 OK
Date: Mon, 20 Jun 2005 21:32:59 GMT
Server: Apache/2.0.49 (Fedora)
X-Powered-By: PHP/4.3.4
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

Campo1 opcion 1<br>
Campo2 opcion 2<br>
```

En el navegador (Fig. 9):



## Comandos HTTP

Lista de comandos, descripción y versión a partir de la cuál está disponible.

Método	Versión	Descripción
GET	HTTP/0.9	Trae un documento
HEAD	HTTP/1.0	Trae la parte de cabecera de un documento
POST	HTTP/1.0	Envía datos a un servidor
PUT	HTTP/1.1	Solicita al servidor guarde la información que se envía
DELETE	HTTP/1.1	Borra un archivo en el servidor
TRACE	HTTP/1.1	Destinado a fines de depuración y seguimiento
CONNECT	HTTP/1.1	Reservado para los proxys a fin de crear un túnel
OPTIONS	HTTP/1.1	Lista las opciones de un determinado recurso

## Códigos de Respuesta HTTP

Asimismo a cada petición HTTP el servidor contesta con un mensaje de estado, que consta de un número de tres cifras, interpretando este código podemos saber que está ocurriendo con el servidor.

Genéricamente los mensajes de respuesta están agrupados de la siguiente forma:



- **1xx** indica un mensaje informativo solamente (100 continue)
- **2xx** indica que algo ha ocurrido (200 ok)
- **3xx** redirige al usuario a otra URL
- **4xx** indica un error en el cliente (404 host not found)
- **5xx** indica un error en el servidor (500 Internal Server Error)
- **6xx** indica un error en el servidor (errores no estándar)
- 

A continuación a un sumario de los códigos de respuesta del protocolo HTTP y que significa cada código; para información más detallada consulte la sección 10 del RFC 2616 (<http://libraries.ucsd.edu/about/tools/rfc2616-10.html>)

- **1xx: Informativo.** Estos códigos indican una respuesta provisional que debería ser continuada por otra respuesta.
  - 100 Continue El servidor no ha rechazado la parte inicial de la petición y el cliente puede continuar con las siguientes peticiones.
  - 101 Switching Protocols El servidor acepta la petición del cliente de cambiar protocolos.
- **2xx Suceso.** Esta categoría de códigos de respuesta indica que la petición del cliente se ha recibido correctamente, comprendida y aceptada.
  - 200 OK La petición ha tenido éxito.
  - 201 Created La petición ha sido cumplida y como resultado un nuevo recurso ha sido creado.
  - 202 Accepted La petición ha sido aceptada para su procesamiento, pero el mismo aún no ha sido completado.
  - 203 Non-Authoritative Information. Las cabeceras devueltas no son definitivas, pero han sido generadas a partir de una copia local o de un tercero.
  - 204 No Content El servidor ha cumplido su petición pero no necesita devolver ninguna salida.
  - 205 Reset Content El servidor ha cumplido su petición y el cliente debe resetear la vista del documento (En principio para lograr un ingreso de datos más fácil).
  - 206 Partial Content El servidor ha cumplido su petición GET parcial para el recurso indicado.
- **3xx: Redirección.** Estos códigos de respuesta indican que el cliente será redirigido a otra URI, o que otra acción más deberá ser cumplida por el cliente para que la petición se cumpla.

- 300 Multiple Choices Hay muchos recursos que machean con la petición, el cliente será redirigido a uno de ellos.
- 301 Moved Permanently El recurso pedido se ha movido a una URI nueva permanentemente.
- 302 Found El recurso pedido reside en forma temporal en una nueva URI, hasta que la redirección sea alterada utilice la URI vieja.
- 303 See Other La respuesta a la petición puede que se encuentre bajo una nueva URI y deberá ser obtenida de allí.
- 304 Not Modified El documento no ha sido modificado desde la última petición y la copia local cacheada puede ser usada.
- 305 Use Proxy El recurso pedido debe ser accedido a través de un proxy.
- 306 (Unused) Usado en versiones anteriores del protocolo.
- 307 Temporary Redirect El recurso pedido reside en forma temporal bajo una URI diferente, hasta que la redirección sea alterada, continúe utilizando la URI actual.
- 4xx: Errores del Cliente. Estas respuestas indican que el cliente ha tenido un error.
  - 400 Bad Request La petición no puede ser comprendida por el servidor debido a una mala sintaxis.
  - 401 Unauthorized La petición requiere autenticación.
  - 402 Payment Required Reservado para futuro uso.
  - 403 Forbidden El servidor comprendió su petición, pero rechaza el cumplirla, típicamente por un problema de permisos de acceso.
  - 404 Not Found El servidor no ha encontrado nada que corresponda a la URI pedida.
  - 405 Method Not Allowed El método de petición no está permitido.
  - 406 Not Acceptable El cliente no es capaz de manejar la respuesta.
  - 407 Proxy Authentication Required. El código es similar al 401 pero indica que el cliente se debe autenticar primero en el proxy.
  - 408 Request Timeout El cliente no produjo una petición en el tiempo que el servidor espera.
  - 409 Conflict La petición no pudo ser completada por un conflicto con el estado actual del recurso.
  - 410 Gone El recurso pedido no está disponible en el servidor y no hay una dirección de redirección disponible.

- 411 Length Required El servidor no acepta la petición sin una cabecera Content-Length definida
- 412 Precondition Failed La precondition dada por una o más cabeceras de la petición falló cuando fue probada en el servidor.
- 413 Request Entity Too Large. El servidor está rechazando procesar la petición porque la entidad requerida es más larga que lo permitido por el servidor.
- 414 Request-URI Too Long. El servidor rechaza la petición porque la URI especificada es muy larga.
- 415 Unsupported Media Type. El servidor está rechazando la petición porque el cliente no soporta el formato de la respuesta.
- 416 Requested Range Not Satisfiable. El servidor no puede encontrar el rango especificado por la cabecera Range.
- 417 Expectation Failed El servidor no puede encontrar la expectativa pasada a través del Expect header.
- 5xx: Errores de Servidor. Estas respuestas indican casos en los cuáles el servidor no puede determinar cuál es el problema por el cuál no puede completar la petición.
  - 500 Internal Server Error El servidor encontró una condición inesperada por la cuál no puede completar la petición.
  - 501 Not Implemented El servidor no soporta la funcionalidad requerida en la petición.
  - 502 Bad Gateway El servidor es un gateway o un proxy y recibió una respuesta inválida del servidor de destino.
  - 503 Service Unavailable El servidor está temporalmente incapacitado para manejar la petición por una sobrecarga temporaria mantenimiento.
  - 504 Gateway Timeout El servidor es un gateway o un proxy y no recibe respuesta en tiempo del servidor de destino.
  - 505 HTTP Version Not Supported. El servidor no soporta o rechaza soportar la versión de HTTP requerida.
- 6xx: Errores Internos. Estas respuestas indican un error en el software del servidor (estos códigos de error no son oficiales y son usados reportar problemas).
  - 600 Malformed URI El programa no es capaz de parsear el URI
  - 601 Connection Timed Out. El servidor no ha respondido antes que se produzca el time-out de la conexión.

- 602 Unknown Error Algún error ha sido encontrado contactando el-servidor, creando la respuesta o parseando la respuesta.
- 603 Could Not Parse Reply La respuesta desde el servidor no puede ser parseada.
- 604 Protocol Not Supported. La URI utiliza un protocolo no soportado.

## Referencias

- World Wide Web Consortium. <http://www.w3c.org>
- W3Schools. <http://www.w3schools.com>