INAP
INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

Curso online: **Instalación, Configuración y Administración de Apache + Tomcat**

Módulo 2. Apache Web Server

Capítulo 5. Administración de Apache Web Server

Autores

Janine García Morera

Alexandra López de la Oliva Portugués

Julio Villena Román

Octubre de 2014

Índice de contenidos

CAPÍTULO 5 ADMINISTRACIÓN DE APACHE WEB SERVER

5.1. Introducción

La **administración del servidor Web Apache** se realiza mediante la edición y modificación del fichero de configuración **`conf/httpd.conf`**. La parametrización se realiza mediante el uso y modificación de las directivas de Apache.

Apache no dispone de una GUI para realizar la administración de forma gráfica, aunque si hay herramientas, principalmente comerciales, que facilitan el trabajo.

Los parámetros de Apache pueden afectar a toda la instancia (parámetros globales), al servidor que escucha por defecto (servidor principal) o a los hosts virtuales. Cada módulo puede tener sus propias directivas.

El módulo de multiproceso se define en tiempo de compilación y no puede ser modificado. Hay MPM específicos de determinados sistemas operativos, como MPM_Winnt para Windows.

Una de las características más potentes y más utilizadas de Apache es la definición de hosts virtuales. Gracias a los hosts virtuales se puede hacer funcionar más de un sitio web en una sola máquina de manera transparente al usuario.

Otro tema importante que un administrador debe conocer son los códigos de error. Cada petición HTTP recibe en su respuesta un código de estado de la petición, incluso aunque la respuesta sea correcta. Conocer estos códigos puede ser útil para un administrador cuando quiere hacer un seguimiento del funcionamiento de su servidor.

5.2. Objetivos

- Conocer algunas de las herramientas de administración existentes.
- Saber cómo parametrizar y configurar un servidor Apache.
- Conocer las directivas más importantes y su funcionamiento.
- Saber definir hosts virtuales basados en nombre y basados en IPs.
- Conocer los códigos de error más comunes.

5.3. Herramientas de Administración

La parte más importante de la administración de un servidor web incluye la instalación, configuración y monitorización de diferentes servidores.

La integración de los diferentes módulos con el servidor puede ser a veces una tarea complicada, por lo que se han creado algunas herramientas como **Apache Toolbox** (<www.apachetoolbox.com>) que permiten realizar la compilación e integración de los módulos adicionales con el servidor mediante menús.

En lo que se refiere a la configuración del servidor, Apache no tiene una GUI de administración o configuración, sino que se usan ficheros de configuración en modo texto. Para facilitar este trabajo existen herramientas gráficas Open Source que facilitan este trabajo, como por ejemplo **Webmin** (<www.webmin.com>). Webmin es una interfaz de administración de sistemas Unix y Linux basada en Web, y tiene un módulo que facilita la configuración de un servidor Apache instalado en el mismo sistema.

Algunos sistemas operativos, como Red Hat, tienen su propia herramienta de configuración de Apache.

También hay herramientas comerciales que facilitan la configuración y administración de Apache, como **ApacheConf** y **Apache Commander** (<www.apache-gui.com>).

La administración de Apache se realiza mediante la parametrización a través de sus ficheros de configuración. En la versión actual (la 2.4) el fichero principal de configuración se llama **`httpd.conf`**, aunque puede haber varios (que son cargados en tiempo de startup desde `httpd.conf`) y prácticamente todas las herramientas de administración se limitan a hacer una edición controlada y formateada del fichero de configuración. En un epígrafe posterior se analiza el contenido del fichero `httpd.conf`.

Conceptos importantes	
Herramientas de configuración de Apache	Herramientas gráficas Open Source: Webmin
	Herramientas comerciales: ApacheConf y Apache Commander.

5.4. Administración Básica de Apache

5.4.1. Estructura de directorios Apache

Una vez instalado el Apache, en el directorio raíz de la instalación, se encontrarán los siguientes directorios:

- **`bin`**: Ficheros ejecutables del Apache.
- **`cgi-bin`**: Contiene los scripts CGI para servir contenido de manera dinámica.
- **`conf`**: Ficheros de configuración del servidor.
- **`error`**: Ficheros con los mensajes de error del servidor, en varios lenguajes.
- **`htdocs`**: Directorio raíz por defecto del servidor. Es dónde se guardan las páginas Web.
- **`icons`**: Directorio donde se encuentran los iconos que utiliza el servidor (entre otras cosas para mostrar estructuras de directorios).
- **`include`**: Contiene los ficheros de cabecera.
- **`lib`:** Contiene las librerías necesarias para compilar los módulos de Apache, como por ejemplo APR.
- **`logs`**: Directorio donde se almacenan los registros de acceso y errores del servidor.
- **`manual`**: Directorio donde se encuentra el manual del Apache.

- **`modules`**: Conjunto de módulos que forman parte de la distribución de Apache.

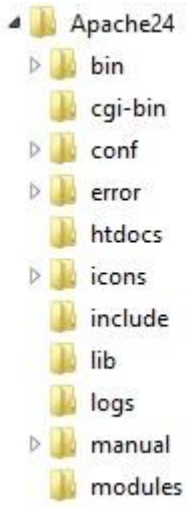

Figura 2.5.1: Estructura de directorios de Apache

5.4.2. Configuración de Apache

La configuración de Apache se realza mediante directivas de configuración escritas en ficheros de texto que se leen al iniciar el servidor web, el fichero de configuración principal se encuentra dentro de la carpeta **`conf`**, con el nombre **`httpd.conf.`** También se utilizan los ficheros de configuración descentralizados que son aquellos con nombre **`.htaccess`** y que se pueden encontrar dentro de cualquier directorio que se encuentre dentro del servidor.

El fichero principal de configuración es `conf/httpd.conf`

Los valores de las directivas que se encuentran dentro de un fichero `.htaccess`, prevalecen frente a los valores de configuración especificados dentro del fichero `httpd.conf`. Asimismo, las directivas de un directorio sobrescribirán a las de su directorio padre.

5.4.3. ¿Qué son las directivas?

Las directivas son instrucciones al servidor Apache para controlar su funcionamiento. Una directiva consiste en un nombre seguido de uno o más argumentos. Por ejemplo:

```
ServerAdmin webmaster@mihost.midominio
```

Las directivas se pueden modificar en los ficheros de configuración de Apache. El ejecutable de Apache lee las directivas de este fichero al arrancar.

Si se ha instalado Apache en Windows y alguno de los directorios en la estructura de instalación tiene blancos en su nombre, todas las referencias a caminos absolutos en el fichero

de configuración deberán ir entre comillas e incluir la letra que especifica el drive en que está instalado. No es necesario, sin embargo, utilizar backslah (\) como delimitador. Por ejemplo:

```
ServerRoot "C:/Archivos de Programa/Apache2.4"
```

es equivalente a

```
ServerRoot "C:\Archivos de Programa\Apache2.4"
```

Pero la siguiente definición fallaría ya que considera cada cadena entre espacios como un parámetro distinto:

```
ServerRoot C:\Archivos de Programa\Apache2.4
```


Como norma general, realizar la instalación en directorios sin espacios en el nombre y utilizar siempre comillas para referenciar los caminos absolutos.

En nuestro ejemplo de instalación en Windows:

```
ServerRoot "c:/Apache24"
```

Conceptos importantes	
Configuración de Apache	Las configuraciones del servidor residen dentro de dos ficheros, el de configuración principal que se encuentra dentro de la carpeta `conf`, con el nombre `httpd.conf`, o dentro de un fichero con el nombre `.htaccess` que se puede encontrar dentro de cualquier directorio que se encuentre dentro del servidor.
¿Qué son las directivas?	Las directivas son instrucciones al servidor Apache para controlar su funcionamiento. Una directiva consiste en un nombre seguido de uno o más argumentos.

5.5. El fichero httpd

5.5.1. Introducción

El fichero **`httpd.conf`** es el fichero principal de configuración del Apache, se encuentra dentro del directorio `conf`, en el directorio de instalación del Apache.

En primer lugar hay que destacar que el fichero está dividido en tres secciones, que son:

1. **Parámetros globales.** Las directivas incluidas en esta sección afectan al funcionamiento del servidor Apache en su totalidad. Es el caso del número de peticiones concurrentes que el servidor puede gestionar o dónde encuentra sus ficheros y directorios. Se incluyen las directivas de los Módulos de Multiprocesamiento (MPMs) que controlan cómo reacciona el servidor ante variaciones en su carga en función de qué MPMs están instalados y el soporte al Dynamic Shared Object (DSO), que es el mecanismo utilizado para añadir módulos externos al servidor.

2. **Directivas del servidor principal**. En esta sección se dan valor a las directivas usadas por el servidor principal Apache. Se incluyen el nombre del servidor, el puerto por el que se aceptan las peticiones y otras.
3. **Host Virtuales.** Contiene las mismas directivas que la sección del servidor principal. Se utilizan para soportar servidores virtuales en un único sistema físico. Estas directivas sobrescriben las de la sección anterior para el host virtual definido en ella.

En el fichero se encuentran todos los parámetros de funcionamiento del Apache. Algunos parámetros son generales para la instalación y funcionamiento del Apache. Muchos otros de los parámetros se pueden configurar independientes para un conjunto de directorios y/o ficheros. En estos casos los parámetros se encuentran ubicados dentro de secciones donde se indica el ámbito de aplicación del parámetro.

Al hacer cualquier modificación en el fichero httpd.conf se recomienda hacer un chequeo posterior ejecutando `httpd -t` antes de reiniciar Apache.

```
C:\Apache24\bin>httpd -t
Syntax OK
```

Figura 2.5.2: Ejecución del comando `httpd -t`

El fichero **`httpd.conf`** no tiene por qué ser el único fichero de configuración: las directivas de configuración se pueden distribuir entre varios ficheros, realizando agrupaciones lógicas (por ejemplo, fichero de configuración de servidores virtuales, de configuración de SSL, etc), pero debe haber una cláusula `include` en el fichero `httpd.conf` que referencie y cargue los ficheros secundarios. Desde la versión 2.2.2 de Apache, se incluye en la estructura de directorios un subdirectorio `conf/` extra en el que hay una serie de ficheros de configuración de componentes (nombre de defecto `httpd_componente.conf`, por ejemplo `httpd_ssl.conf`) que se pueden activar descomentando la cláusula `include` correspondiente del fichero de configuración principal `httpd.conf`.

Además, existe la posibilidad de modificar las especificaciones de las directivas a nivel de directorio, usando para ello ficheros con un nombre específico (normalmente `.htaccess`) que se ubican en el directorio cuyas directivas se quieren modificar o en cualquiera de sus directorios padre.

5.5.2. Parámetros globales

En el fichero de configuración `httpd.conf` no hay una separación entre las directivas que especifican los parámetros globales y las del servidor principal. Por ello es necesario conocerlas para saber su ámbito de aplicación. Las directivas definidas como parámetros globales afectan a toda la instancia del servidor, incluyendo servidor principal y hosts virtuales, y no pueden ser modificadas en ninguno de estos dos ámbitos. Las principales las vamos a analizar en los siguientes apartados.

5.5.2.1 DIRECTIVAS DE INSTALACIÓN

- **`ServerRoot`**: define la raíz del árbol de directorios en el que se ubican los ficheros de configuración del servidor, los ficheros de errores y los ficheros de log. Esto es el directorio donde se instala Apache:

```
ServerRoot /path_apache
```

No poner nunca slash (/) al final de los parámetros en que se especifique un directorio:

```
ServerRoot /path_apache
```

y no:

```
ServerRoot /path_apache/
```

Si el `ServerRoot` se ubica sobre un directorio montado en un sistema de ficheros NFS, la directiva `LockFile` debe apuntar a un fichero en disco local.

En nuestro ejemplo de instalación en Windows:

```
ServerRoot "C:/Apache24/"
```

- **`PidFile`**: esta directiva sirve para indicarle a Apache donde puede encontrar el fichero en el que se incluirá el PID (identificador de proceso) del proceso principal de Apache (el proceso principal que se crea al ejecutar apache). Por defecto se ubica en **`{serverroot}/logs/httpd.pid`** y no es necesario definirlo a no ser que haya varias instancias de apache en ejecución compartiendo el mismo software (el mismo `ServerRoot` con distintos ficheros de configuración).

5.5.2.2 SOPORTE DSO (DYNAMIC SHARED OBJECT)

Cuando se genera el ejecutable apache (`httpd`), se compila montando estáticamente una serie de módulos.

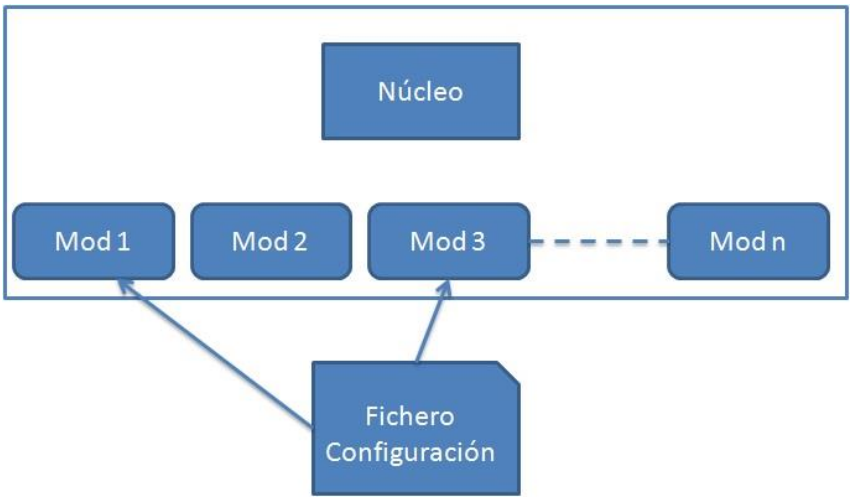

Figura 2.5.3: Generación del `httpd`

Se puede obtener la lista de los módulos linkados estáticamente en el binario `httpd` (o apache) ejecutándolo con el parámetro –l desde la línea de comandos.

```
C:\Apache24\bin>httpd -l
Compiled in modules:
  core.c
  mod_win32.c
  mpm_winnt.c
  http_core.c
  mod_so.c
```

Figura 2.5.4: Ejecución del comando `httpd -l`

Sin embargo, si se quiere utilizar más módulos de los que dispone Apache (directorio modules) o módulos de terceros, no es necesario volver a compilar linkando estáticamente dichos módulos. Se puede reconfigurar Apache fácilmente añadiendo módulos dinámicos compartidos con la directiva **`LoadModule`**. El soporte para cargar módulos individuales se encuentra en el módulo **`mod_so.c`**, que, como hemos visto, está montado estáticamente con el ejecutable `httpd`. Es lo que se llama el soporte de Objetos Compartidos Dinámicos (DSO). El límite de módulos compartidos a usar con un ejecutable Apache se fija en tiempo de compilación (por defecto, 128).

El DSO tiene las siguientes ventajas:

- El servidor es más flexible porque los procesos de servidor pueden ser ensamblados en tiempo de ejecución usando la directiva `LoadModule`.
- El servidor se puede ampliar con módulos de terceros incluso después de la instalación.
- Si un módulo se ensambla estáticamente y luego no se utiliza, está usando recursos del sistema inútilmente. Si se utiliza el módulo dinámico compartido, basta con no cargarlo.

La directiva **`LoadModule`** monta el fichero objeto o librería incluidos como parámetro en el espacio de direccionamiento del servidor y añade la estructura del módulo a la lista de módulos. Por ejemplo:

```
LoadModule nombre_modulo libexec/externo.so
```

Un ejemplo real sería el módulo `dir_module` (este módulo sirve para indicar cuál es el documento que se sirve por defecto en caso de que se reciba una petición que apunte directamente al servidor) :

```
LoadModule dir_module modules/mod_dir.so
```

Se pueden ver los módulos cargados estática y dinámicamente desde la línea de comandos, mediante el parámetro –M (o su equivalente –t –D DUMP_MODULE).

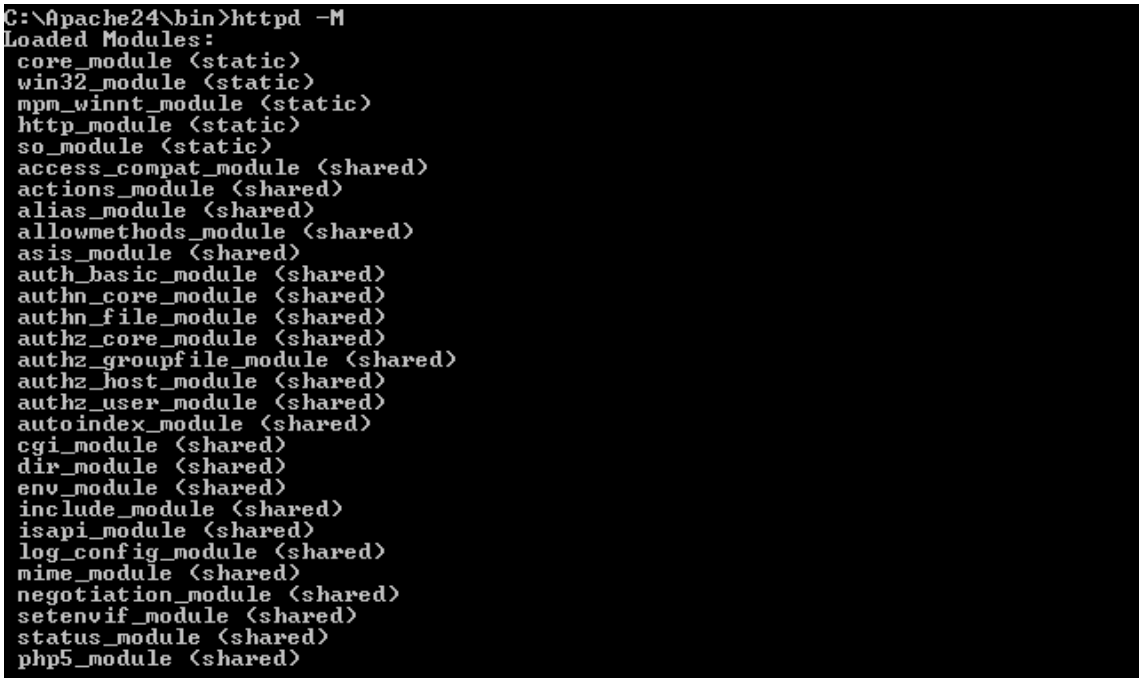

Figura 2.5.5: Ejecución del comando `httpd -M`

Cada módulo tiene sus propias directivas que controla su comportamiento. Estas directivas se incluyen en el fichero de configuración y, en algunos casos, su uso puede provocar errores si el módulo correspondiente no está cargado. Se puede controlar si un módulo está o no cargado mediante la directiva <`IfModule`>. Se usa de la siguiente forma:

```
<IfModule nombre_modulo>
   directiva1_de_modulo
   directiva2_de_modulo
   ...
   directivaN_de_modulo
</IfModule>
```

En nuestro ejemplo, con el módulo `dir_module`:

```
#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

También se puede controlar si un módulo no está cargado, mediante la directiva `IfModule !nombre_modulo`.

5.5.2.3 DIRECTIVAS DE GESTIÓN DINÁMICA DE CARGA

El servidor Apache 2.x se adapta dinámicamente a su carga, manteniendo suficientes procesos o hilos para dar servicio a las peticiones activas, así como unos pocos procesos sobrantes

(spare) para dar servicio a nuevas peticiones. Los parámetros que se deben usar dependen de qué MPM se instale.

Como ya se indicó los **Módulos de Multiproceso (MPMs)** extienden el diseño modular de la tecnología Apache, realizando las funciones más básicas de un servidor web: asignación de puertos de red, aceptación de peticiones y generación de los hijos para dar servicio a las peticiones. Los MPMs permiten a Apache soportar eficientemente una gran variedad de sistemas operativos.

Con los MPMs el servidor puede ser personalizado para adaptarse a las necesidades de un sitio en particular. Por ejemplo, sitios que necesitan de escalabilidad pueden usar un modelo como el Worker MPM, mientras que sitios que requieren estabilidad y compatibilidad con software antiguo pueden usar el Prefork MPM. Sin embargo, no todos los tipos de MPM pueden ser utilizados en todos los Sistemas Operativos.

Los módulos MPM se ensamblan estáticamente con el ejecutable Apache (`httpd`) en tiempo de compilación, y no son descargables dinámicamente. Si se pueden controlar su carga mediante el uso de <`IfModule`>, por lo que se puede definir un fichero httpd.conf genérico para varias instalaciones con distintos MPMs.

1. **Prefork MPM**. Este modelo implementa un servidor basado en procesos. Manipula peticiones en una manera muy similar al comportamiento por defecto de Apache 1.3. Cada proceso puede servir una petición. Es un servidor muy robusto, ya que los procesos no comparten código entre sí, por lo que si un proceso funciona mal el resto no se ven afectados. Sin embargo, el hecho de no compartir código y de que sea necesario un proceso por petición afecta negativamente al rendimiento.

```
<IfModule mpm_prefork_module>
  StartServers                 5
  MinSpareServers              5
  MaxSpareServers             10
  MaxRequestWorkers          150
  MaxConnectionsPerChild       0
</IfModule>
```

2. **Worker MPM**. Este modelo implementa un híbrido de servidores multiproceso y multithreaded para sistemas que soportan threads POSIX. Cada proceso tiene un número fijo de threads. Cuando se recibe una petición, se le pasa a un worker para ser procesado. El servidor ajusta sus cambios en su carga aumentando o disminuyendo el número de procesos. Este servidor escala muy fácilmente pero a costa de la robustez.

```
<IfModule mpm_worker_module>
  StartServers                 2
  MaxRequestsWorkers         150
  MinSpareThreads             25
  MaxSpareThreads             75
  ThreadsPerChild             25
</IfModule>
```

En la documentación de Apache (<httpd.apache.org/docs/current/mod/mpm_common.html>) podemos ver las directivas comunes a ambos módulos, aquí mencionamos las más importantes.

- → **`StartServers`**: indica el número de procesos de servidor que se inician al levantar Apache.

- **`MaxRequestWorkers`**: controla el número de peticiones simultáneas que serán servidas. Cualquier intento de conexión por encima del límite será encolado hasta un número basado en la directiva `ListenBacklog`. La conexión será servida una vez que un proceso hijo de una petición diferente se libere. En anteriores versiones se llamaba `MaxClients` (aún soportado). Su valor por defecto es 256.

  Este parámetro se debería de fijar en función de los recursos disponibles en el sistema, fundamentalmente la memoria. Si el número es demasiado alto, al alcanzar un determinado número de conexiones el sistema comenzará a hacer **swapping**, con lo que el rendimiento del servidor se verá muy degradado.

- **`MaxConnectionPerChild`**: establece el límite de conexiones que un proceso hijo puede atender. Esto evita problemas que ocurrirían después de un uso prolongado del proceso. Después de este límite de conexiones, el proceso hijo morirá. Si `MaxConnectionPerChild` es 0 el proceso nunca expirará. En anteriores versiones se llamaba `MaxRequestPerChild`.

Las siguientes directivas solo se usan para el **Prefork MPM**:

- **`MinSpareServers`**: Es el mínimo número de procesos de servidor que se quedan como sobrantes (**spare**). Si en el funcionamiento del servidor quedan menos procesos inactivos de lo que fija este parámetro, Apache crea nuevos procesos hasta alcanzar dicho número. Si este parámetro es demasiado bajo, y la carga del sistema es alta, la respuesta del servidor será lenta, ya que se estarán creando nuevos procesos continuamente.

- **`MaxSpareServers`**: número máximo de procesos inactivos sobrantes. Si hay más procesos inactivos sobrantes de lo que indica este parámetro, Apache termina los procesos adicionales. Aumentar este número permitirá al servidor responder más rápidamente los cambios en las condiciones de carga. Sin embargo, aumentar el número de procesos de servidor también aumenta la carga del sistema.

Las siguientes directivas solo se usan para el **Worker MPM**:

- **`MinSpareThreads`**: similar a `MinSpareServers`, define el mínimo número de hilos workers que actúan como sobrantes. Si el número de workers inactivos está por debajo de este número, Apache crea otro nuevo proceso de servidor para manejar la carga.

- **`MaxSpareThreads`**: análogo a `MaxSpareServers`, define el máximo número de hilos workers que actúan como sobrantes. Si hay más workers inactivos de lo que indica este parámetro, Apache termina el proceso que ejecuta esos hilos.

- **`ThreadsPerChild`**: este parámetro define el número de workers creado por cada proceso servidor. Es un valor constante.

3. **MPM winnt**: es el MPM para Windows (Windows NT, Windows 2000, Windows 2003, Windows XP …). Es un módulo multithread. Al arrancar, Apache crea un proceso padre (httpd) y un proceso hijo (httpd) que crea todos los hilos que van a servir las peticiones.

```
<IfModule mpm_winnt_module>
  ThreadsPerChild                 150
  MaxConnectionsPerChild            0
</IfModule>
```

- **`ThreadsPerChild`**: este parámetro define el número de threads creado por cada proceso servidor. Es un valor constante.

- **`MaxConnectionsPerChild`**: controla el número de peticiones que un thread puede servir antes de que se le obligue a morir. Esto evita problemas que

ocurrirían después de un uso prolongado de proceso. Si el parámetro se fija a cero, los threads pueden dar servicio a un número ilimitado de peticiones.

5.5.3. Directivas de configuración del servidor principal

Una de las características más importantes y potentes de Apache es la posibilidad de definir varios hosts virtuales en una misma instancia. Cada host virtual puede escuchar por un puerto distinto, tener un nombre diferente o una dirección IP distinta, así como su propio espacio web. Sin embargo, es necesario definir la estructura de un servidor principal, que es el que sirve el contenido si no se definen servidores virtuales o si posteriormente se definen, será el que permita heredar las características básicas que no son modificadas en la definición de un host virtual. Todos los parámetros y directivas del servidor principal pueden ser modificadas para un host virtual determinado: si no se hace así, toman el valor definido para el servidor principal.

5.5.3.1 DIRECTIVAS GENERALES

- **`Listen`**: define el puerto en el cual el servidor escucha las peticiones. Para puertos menores al 1024, se debe ejecutar `httpd` como root (en windows lo ejecuta localsystem). El puerto por defecto es el 80. Esta directiva es obligatoria, aunque se escuche por el puerto de defecto. Por ejemplo:

```
Listen     80
```

  Se pueden definir múltiples directivas listen, permitiendo a Apache responder peticiones en otros puertos. También se pueden definir puertos distintos en cada `<VirtualHost>`.

- **`User / Group`**: estas dos directivas definen el nombre (UID) o grupo (GID) del usuario/grupo propietario del proceso `httpd` en ejecución. Si se quiere ejecutar apache con un usuario o grupo distinto, deberá ejecutarse como root. Durante la inicialización, cambia automáticamente al usuario/grupo definidos. Los valores por defecto son los siguientes:

```
User    #-1
Group   #-1
```

  Se pude ejecutar apache con un usuario que no sea root, siempre y cuando los permisos de los ficheros y directorios estén correctamente asignados.

- **`ServerAdmin`**: la dirección de correo del administrador del Servidor web:

```
ServerAdmin webmaster@mihost.midominio
```

  Esta directiva aparece en algunas páginas generadas, tales como páginas de error.

- **`ServerName`**: permite fijar un nombre para el servidor que puede ser diferente de su hostname. Puede ser una dirección IP.

```
ServerName miservidor.midominio
```

  Si el servidor está escuchando por un puerto no estándar, se debe incluir el puerto en la directiva `ServerName`. Por ejemplo:

```
Listen 8080
ServerName miservidor.midominio:8080
```

En el caso de que el servidor no tenga registrado un nombre DNS, se introducirá aquí su IP.

- **DocumentRoot**: define el directorio que contiene los ficheros y directorios que Apache pone a disposición de los usuarios. Por ejemplo:

```
DocumentRoot    "/path_apache/htdocs"
```

En nuestro ejemplo de instalación en Windows:

```
DocumentRoot "c:/Apache24/htdocs"
```


No acabar los directorios con slash (/)

- **AccessFileName**: esta directiva indica el nombre de un fichero que permite redefinir el control de acceso a un directorio específico. El valor por defecto es `.htaccess`. Se puede cambiar el nombre por razones de seguridad, pero en cualquier caso es preciso protegerlo mediante la directiva `Files`, como se explica más adelante.

  Hay que resaltar que el servidor busca este fichero en cada acceso no sólo en el directorio al que se refiere la URL, sino también en los superiores. Por ejemplo, si una URL se refiere a un documento que se encuentra en el directorio `/var/www/html/`, el servidor buscará el fichero `/.htaccess`, `/var/.htaccess/`, `/var/www/.htaccess` y `/var/www/html/.htaccess`. Si no se va a utilizar esta funcionalidad, es conveniente desactivarla mediante la directiva `AllowOverride`.

- **TypesConfig**: esta directiva asigna el nombre y ubicación del archivo de configuración mime (asignación de extensiones y aplicaciones). Por defecto su valor es `conf/mime.types`.

- **DefaultType**: esta directiva no tiene otro efecto que emitir warnings si su valor no es none. En versiones anteriores esta directiva asignaba el tipo mime por defecto a todos los ficheros cuyo tipo no pueda ser determinado por la extensión. Esta directiva ha sido deshabilitada pero por compatibilidad con versiones anteriores se ha mantenido con el valor por defecto none. Se recomienda el uso del fichero de configuración mime.types y la directiva `AddType` para este tipo de asignaciones a ficheros.

5.5.3.2 DIRECTIVAS DE CONTENEDORES

Las directivas se aplican dentro de un ámbito. Si una directiva no aparece dentro de un ámbito se aplica a todo el servidor Apache.

Las directivas de contenedores sirven para delimitar este ámbito y se utilizan para aplicar grupos de otras directivas, principalmente de control de acceso, de forma delimitada a determinados directorios, URL o ficheros.

- **Directiva de contenedor <Directory> ... </Directory>**: esta directiva provee control de acceso por directorio. Se usa para agrupar varias directivas que se aplican sólo al directorio indicado y a sus subdirectorios.

  Se pueden definir los servicios y características que están explícitamente permitidas o prohibidas en los directorios a los que se puede acceder. El primer paso es configurar el directorio raíz con un conjunto de permisos muy restrictivo:

```
<Directory />
   Options  FollowSymlinks
   AllowOverride  None
   Require all denied
```

```
</Directory>
```

De esta manera se impide el acceso por defecto a cualquier directorio del servidor web.

En las siguientes entradas se deben permitir específicamente los permisos que se quieran activar. El siguiente paso es configurar los permisos para el directorio **`DocumentRoot`** (el que sirve los documentos) y sus subdirectorios.

```
<Directory "/path_apache/htdocs">
   Options Indexes FollowSymLinks MultiViews
   AllowOverride  None
  Require all granted
</Directory>
```

Existe la directiva equivalente `<Directory>` que permite el uso de expresiones regulares.

A continuación explicamos directivas que son específicas de la directiva de contenedor `<Directory>`:

- **`Options`**: esta directiva controla las características que están disponibles en una directiva `<Directory>` particular. Toma un argumento, **option**, que se puede fijar a los siguientes valores (entre otros):

Argumento	Descripción
None	No se activan características extras
All	Esta es la opción por defecto que permite todas las opciones excepto `MultiViews`.
FollowSymLinks	El servidor sigue links simbólicos que puede haber en este directorio.
Indexes	Si se solicita una URL que mapea a un directorio (no a un fichero) y no hay en ese directorio ningún fichero `DirectoryIndex` (por ejemplo, `index.html`), entonces el servidor devuelve una lista formateada del contenido del directorio.
Includes	En el lado del servidor los `Includes` de `mod_include` están permitidos.
IncludesNOEXEC	En el lado del servidor los Includes están permitidos, pero los comandos exec cmd y `exec-cgi` están deshabilitados.
MultiViews	Se permite contenido negociado `MultiViews`. Una petición por un fichero, como puede ser `index.html`, puede ser sustituido por extensiones negociadas siguiendo la petición base: `index.html.es`, `index.html.en`, etc., en este caso en función de las preferencias de lenguaje seleccionadas en el navegador.
SymLinksIfOwnerMatch	El servidor solo sigue los links simbólicos en los que el directorio o fichero a los que apunta el enlace son propiedad del mismo usuario que el link.
ExecCGI	Se permite la ejecución de CGI scripts en ese directorio usando `mod_cgi`

Normalmente, si se pueden aplicar múltiples directivas `Options` a un directorio, entonces la más específica se aplica y las demás se ignoran; las opciones no se

fusionan. Sin embargo, si todas las opciones en la directiva `Options` van precedidas de un símbolo + o -, las opciones se fusionan. Cualquier opción precedida de un + se añade a las opciones en ese momento activas, y las opciones precedidas de un - se quitan de las activas en ese momento.

Por ejemplo, sin ningún símbolo + o -:

```
<Directory /web/docs>
   Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec/>
 Options Includes
</Directory>
```

Entonces solo `Includes` tendrá efecto para el directorio `/web/docs/spec`, ya que la segunda anula la primera. Sin embargo, si la segunda directiva `Options` usara un símbolo + y otro -:

```
<Directory /web/docs>
   Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec/>
   Options +Includes -Indexes
</Directory>
```

Entonces las opciones `FollowSymLinks` e `Includes` estarán activadas para el directorio `/web/docs/spec` (el uso de `-IncludesNOEXEC` o `-Includes` desactiva server-side includes completamente independientemente de la configuración anterior).

El comportamiento por defecto en ausencia de ninguna configuración es de All.

- **`AllowOverride`**: esta directiva define qué directivas declaradas en un fichero especificado por `AccessFileName` (normalmente `.htaccess`) ubicado en el directorio o cualquiera de sus directorios padre pueden modificar información de acceso previamente declaradas a nivel general. Su valor por defecto es None (antes de la versión 2.3.9 su valor por defecto era All). Cuando esta directiva se fija a None, entonces el fichero `.htaccess` se ignora completamente. Si se fija a All, cualquier directiva incluida en un fichero `.htaccess` es tenida en cuenta. Esta directiva sólo es válida dentro de una sección `<Directory>`. También puede tomar los siguientes valores:

  i) **`AuthConfig`**: permite el uso en el fichero `.htaccess` de directivas de autorización: `AuthName`, `AuthType`, `AuthUserFile`, `AuthGroupFile`, `Require`,…

  ii) **`FileInfo`**: permite el uso en el fichero .htaccess de directivas de control de tipos de documentos: `AddEncoding`, `AddLanguage`, `AddType`, `DefaultType`, `ErrorDocument`…

  iii) **`Indexes`**: permite el uso en el fichero .htaccess de directivas que controlan la indexación de directorios: `AddIcon`, `DirectoryIndex`, `IndexOptions`…

  iv) **`Limit`**: permite el uso en el fichero `.htaccess` de directivas que controlan el acceso al host: `Allow`, `deny` y `order`.

  v) **`Options`**: permite el uso en el fichero .htaccess de directivas que controlan características específicas de directorio: `Options` y `XBitHack`

- **Directiva de contenedor `<Files>`…`</Files>`**: esta directiva provee control de acceso por nombre de fichero. Es similar a la directiva `<Directory>`, y se usa para encerrar un

grupo de directivas que se aplican a un objeto con un nombre de fichero como el especificado. Se pueden utilizar comodines, como por ejemplo:

```
<Files ~ "^\.ht">
   Require all denied
</Files>
```

Este grupo de directivas impide el acceso a todos los ficheros cuyo nombre comienza por .ht.

Existe la directiva equivalente `<FilesMatch>` que permite el uso de expresiones regulares.

- **Directiva de contenedor `<Location>`…`</Location>`**: provee control de acceso usando una URL. Es similar a las directivas anteriores, y se usa para encerrar un grupo de directivas que sólo se aplican a la URL indicada. Esta directiva se puede usar para referirse a recursos externos al servidor.

  Por ejemplo:

```
<Location /status>
   SetHandler server-status
   Require ip  192.23.12.5
</Location>
```

  Este conjunto de directivas permite el acceso a la URL `/status` (`http://miservidor/status`) exclusivamente desde la dirección IP 192.123.12.5 y lo prohíbe para el resto.

  Existe la directiva equivalente `<LocationMatch>` que permite el uso de expresiones regulares.

Cuando se quiera aplicar el ámbito a objetos que residan en el sistema de ficheros se usará `<Directory>` o `<Files>`. En este caso es recomendable no usar `<Location>` ya que varias URLs podrían corresponderse con una misma ubicación en el sistema de ficheros, haciendo que la restricción pueda ser evitada. Se usará `<Location>` cuando se desee aplicar restricciones sobre objetos que no residan en el sistema de ficheros.

5.5.3.3 MECANISMOS DE AUTORIZACIÓN Y SEGURIDAD

1. **Directivas de control de acceso**: se controla el acceso a un determinado recurso (directorio, URL) mediante el nombre del host o la dirección IP del cliente. Cuando se solicita el acceso, Apache comprueba que el ordenador cliente que intenta acceder tiene permisos de acceso a dicho recurso o no.

   En la versión 2.4 se introduce el nuevo módulo `mod_authz_host` para el control de acceso por host o ip. En anteriores versiones, el control de acceso se realizaba mediante las directivas `order`, `deny` y `allow` (por compatibilidad se ofrece el módulo `mod_access_compat` aunque este ha sido deprecado).

   El control de acceso utiliza la directiva `Require`.

   - **`Require`**: se utiliza para especificar si un usuario está autorizado o no a acceder a cierto recurso. El acceso se controla por hostname, IP o rango de IP o por otras características de la petición del cliente capturada en variables de entorno. Cuando se especifica más de un valor (varias direcciones IP, por ejemplo) se separan con espacios.

     Puede tener los siguientes valores:

Valor	Ejemplo	Descripción	Anteriores versiones
All	Require all granted	Permite el acceso a todos los hosts	Allow from all

	Require all denied	No permite el acceso a ningún host	Deny from all
Nombre del host	Require host server.map.es	Permite el acceso solo a server.map.es	Allow from server.map.es
	Require not host server.map.es	No permite el acceso a server.map.es	Deny from server.map.es
Nombre del dominio	Require host map.es	Cualquier servidor del dominio map.es (se servirá server.map.es)	Allow from map.es
	Require not host map.es	Ningún host del dominio map.es	Deny from map.es
Dirección IP	Require ip 10.2.32.1	Solo el servidor 10.2.32.1	Allow from 10.2.32.1
	Require not ip 10.2.32.1	No al servidor 10.2.32.1	Deny from 10.2.32.1
Dirección de subred	Require ip 10.2.32	Toda la subred 10.2.32.x	Allow from 10.2.32
	Require not ip 10.2.32	Ningún host perteneciente a la subred 10.2.32.x	Deny from 10.2.32
Ip/máscara	Require ip 10.1.0.0/255.255.0.0	Permite acceso con Ip 10.1.0.0 y máscara 255.255.0.0	Allow from 10.1.0.0/255.255.0.0
	Require not ip 10.1.0.0/255.255.0.0	No acceso a Ip 10.1.0.0 y máscara 255.255.0.0	Deny from 10.1.0.0/255.255.0.0
Rango de Ips	Require ip 10 172.20 192.168.2	Todos los clientes con una Ip del rango	Allow from 10 172.20 192.168.2
	Require not ip 10.172.20 192.168.2	Ningún cliente con una Ip del rango	Deny from 10 172.20 192.168.2
local	Require local	Permite acceso local si se cumple algula de estas condiciones: ➢ La Ip del cliente coincide con 127.0.0.0/8 ➢ La Ip del cliente es ::1. ➢ Coincide la Ip del cliente y del servidor.	
	Require not	No permite acceso	

	local	local	

Para restringir el acceso se utiliza `Require not`. Ej: `Require not host server.map.es`.


En versiones anteriores era necesario utilizar la directiva `Order` para controlar en qué orden se aplicaban las directivas `allow` y `Deny`.

Con la nueva configuración para que pudieran acceder todos los hosts del dominio example.org y denegar el acceso a cualquier otro.

```
Require host example.org
```

Anteriormente

```
Order Deny, Allow
Deny from all
Allow from example.org
```

- **<Limit>…</Limit>**: esta directiva permite usar directivas de control de acceso (las anteriormente definidas) para que se apliquen sólo a los métodos http que se especifiquen en el cuerpo de la directiva. Estos métodos pueden ser GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK Y UNLOCK. Por defecto, cuando se usan directivas de control de acceso, todos los métodos quedan restringidos (o permitidos) a no ser que se especifiquen en una cláusula `<Limit>`.

  Por ejemplo:

```
<Limit GET POST OPTIONS PROPFIND>
   Require all granted
</Limit>
```

  Este ejemplo, permite el acceso a todos pero solo a los métodos indicados.

- **<LimitExcept>…</LimitExcept>**: su funcionalidad es la contraria a la directiva `<Limit>`: su ámbito se aplica a todos los métodos que no se especifican como parámetro.

2. Autentificación y autorización de usuarios.

Los módulos de autentificación de usuarios permiten proporcionar o denegar el acceso a un usuario o grupo de usuarios a diferentes partes del sitio web.
Para llevar a cabo el proceso se necesitará:

- Especificar el protocolo de autentificación de usuarios:
  - Basic: envía la contraseña en claro (`mod_auth_basic`).
  - Digest: Envía la hash de la contraseña (`mod_auth_digest`).

  Cualquier de estos módulos se utilizará conjuntamente con `mod_authz_core` y `mod_authn_core`, qué son los que contienen las directivas comunes a ambos.
- Indicar el proveedor de autentificación (que es el que establece el método de almacenamiento de las contraseñas en el servidor): en un fichero (`mod_authzn_file`), con un servicio ldap (`mod_authnz_ldap`), en una base de datos (`mod_auth_dbd`)…
- Indicar quiénes están autorizados a acceder al recurso. Qué usuarios (`mod_authz_user`), qué grupos (`mod_authz_group`)...

Puede verse toda la información: <httpd.apache.org/docs/2.4/es/howto/auth.html**>**

- A continuación vamos a ver las directivas más básicas para configurar la autentificación (mecanismo basic, almacenamiento de contraseñas en un fichero y autorización a un conjunto de usuarios y grupos).

  - **`AuthName`**: asigna un nombre para que aparezca en la pantalla de identificación cuando se accede a un recurso protegido. Se utiliza para identificar dicho recurso.

  - **`AuthType`**: esta directiva permite seleccionar el tipo de autentificación de usuario para un recurso determinado. Actualmente sólo se contempla la autentificación BASIC (nombre de usuario y contraseña) y la DIGEST (basada en el uso del algoritmo MD5 y no disponible para todos los navegadores).

  - **`AuthBasicProvider`**: por defecto, file.

  - **`AuthUserFile`**: esta directiva referencia el fichero (con path absoluto) que contiene los nombres de usuario y las claves usadas en la autentificación básica. El fichero de claves se puede generar y modificar mediante el uso del comando `htpasswd` del directorio `bin` de Apache. El siguiente ejemplo crea el fichero `.htpasswd` con el usuario de nombre usuario1 y la clave que se suministra.

```
C:\Apache24\bin>htpasswd -c ..\passwords\.htpasswd usuario1
New password: ********
Re-type new password: ********
Adding password for user usuario1
```

Figura 2.5.6: Creación del fichero `.htpasswd`

```
AuthUserFile "C:/curso/ficheros/.htpasswd"
```

Los ficheros con claves de usuarios deben almacenarse fuera de la estructura del servidor web por razones de seguridad.

  - **`AuthGroupFile`**: especifica el fichero (con path absoluto) que contiene una lista de grupos de usuarios para realizar autentificación básica. El formato del fichero será:

```
Grupo: usuario1 usuario2 usuario3… usuarioN
```

  - **`Require`**: permite definir qué usuarios o grupos de usuarios pueden acceder a un determinado recurso protegido. Puede contener un nombre de uno o varios usuarios, un nombre de uno o varios grupos o bien cualquier usuario válido (que haya realizado correctamente la autentificación).

```
Require user usuario1 usuario2 usuario3
Require group grupo1 grupo2 grupo3 grupo4
Require valid-user
```

El siguiente conjunto de directivas impide el acceso al recurso `/direccion1` a todos los usuarios que no se identifiquen correctamente. Los usuarios y claves se almacenan en `C:\ficheros\.htpasswd`.

```
<Location /direccion1>
AuthName "Recurso Protegido"
AuthType Basic
AuthUserFile "C:/ficheros/.htpasswd"
Require valid-user
</Location>
```

  - **`Satisfy`**: En una configuración básica de autentificación en la que se utilizan `allow` (control por IP) y `require` (control por usuario), la directiva `satisfy` permite decirle a Apache si los requisitos de autentificación son suficientes. Puede tomar dos valores: any y all: si es all, Apache sólo entiende que el usuario puede

acceder al recurso si cumple ambos requisitos (acceso por IP con un usuario válido); si es any, Apache entiende que el usuario puede acceder al recurso si cumple uno de los dos requisitos (accede desde un puesto autorizado o bien si se identifica mediante un usuario y clave válidos).

- Otras Directivas:
  - **`AuthMerging`**: controla la manera en que la lógica de autentificación de cada sección se combina con el de las secciones anteriores de configuración. Cuando la autentificación se activa normalmente, se hereda por cada sección de configuración posterior, a menos que se especifique un conjunto diferente de directivas de autentificación. Esta es la configuración por defecto de esta directiva (Off). Sin embargo, en determinadas circunstancias es deseable que la configuración de una sección sea combinada con su predecesora. Para esto hay dos opciones And y Or.

    Cuando una configuración de una sección tiene `AuthMerging And` o `AuthMerging Or` su lógica se combina con la de su predecesora más cercana

    Por ejemplo:

```
<Directory /www/docs>
   Require group alpha
</Directory>.
<Directory /www/docs/ab>
   AuthMerging Or
   Require group beta
</Directory>
<Directory /www/docs/ab/gamma>
   Require group gamma
</Directory>
```

    En este ejemplo solo los usuarios que pertenecen al grupo alpha pueden acceder a `/www/docs`. Los usuarios que pertenezcan al grupo alpha o beta pueden acceder a `/www/docs/ab`. Además, `AuthMerging Off` se aplica para `/www/docs/ab/gamma` por lo que las directivas de autentificación de esa sección sobrescriben a las predecesoras. De esta forma, solo los usuarios del grupo gamma pueden acceder a `/www/docs/ab/gamma`.

5.5.3.4 FICHEROS DE LOG

Todo servidor debe dejar constancia de la actividad que realiza. Los ficheros de logs permite además de diagnosticar anomalías, obtener información sobre el tráfico que procesa el servidor.

Lo recomendable es que a estos ficheros de log solo tenga acceso el usuario con el que se ejecuta el servidor.

Las directivas para controlar los mensajes de error son:

- **`ErrorLog`**: indica la ubicación del fichero de registro de errores. Un host virtual puede tener su propio fichero de errores, pero si no se indica nada en la sección `<VirtualHost>` utiliza el mismo que el servidor principal.

```
ErrorLog logs/error.log
```

- **`LogLevel`**: controla el nivel de traza de los mensajes de error. Los valores posibles son:

Valor	Descripción
Emerg	Emergencia. El servidor no funciona.
Alert	Se debe tomar una acción inmediatamente.
Crit	Condiciones críticas.
error	Condiciones de error.
warn	Condiciones de warning.
notice	Condiciones significativas dentro del funcionamiento normal.
info	Informativo.
debug	Mensajes de debug (máximo nivel de traza).

Fijar el `LogLevel` a un determinado nivel significa que también se registrarán los errores de un mayor nivel de significación. Por ejemplo, si se fija a nivel de error, se registrarán también los errores de nivel crítico, de alerta y de emergencia. Se recomienda fijar como mínimo el nivel crítico.

El servidor genera un mensaje por cada petición HTTP que es procesada. Las directivas para controlar el control de accesos son:

- **`CustomLog`**: esta directiva se utiliza para registrar las peticiones al servidor. El valor debe ser el nombre de un fichero relativo al `ServerRoot`., o bien una tubería (|) seguida del path de un programa que recibe como entradas los registros de log, y el nombre de un formato definido en la directiva `LogFormat`.

```
CustomLog logs/Access_log common
```

- **`LogFormat`**: especifica el formato del fichero de log definido en `CustomLog` y se le da un nombre. Por ejemplo:

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" common
```

En el ejemplo se define un formato con nombre `common`, que luego podrá ser utilizado en una directiva `CommonLog`. Se utiliza para definir el formato el Common Log Format (CLF), cuyas opciones son:

Com.	Significado
%h	El host remoto. Es la dirección IP del host remoto, a no ser que se haya activado la directiva HostnameLookups.
%l	Nombre del usuario remoto para cada conexión. Solo funciona si la máquina cliente ejecuta identd o algo similar.
%u	El user ID del usuario, si está accediendo a un documento protegido por clave.
%t	La fecha y hora de la petición en formato [día/mes/año:hora:minuto:segundo zona]
\"%r\":	La primera línea de la petición recibida del cliente entre comillas.
%>s	Los tres dígitos del código de estado de la petición. El símbolo ">" fuerza a que sea el estado de la última petición y se requiere por si ha sido internamente redirigida.
%b	El número de bytes devueltos al cliente, sin incluir cabeceras.

Una entrada del fichero de log con este formato podría ser la siguiente:

```
10.12.32.123 - - [12/10/2004:09:20:45 -0800]
"GET / HTTP/1.1" 200 658
```

Además del formato CLF existe un segundo formato que se utiliza para accesos "referidos".

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \
\"%{User-agent}i\"" combined
```

```
CustomLog log/acces_log combined
```

El camo `Referer` incluye la información que el cliente envía como URL que contiene la referencia al recurso pedido y el campo `User-agent` contiene la información que el cliente envía para identificarse.

Una entrada del fichero de logo con este formato podría ser la siguiente:

```
10.12.32.123 - - [12/10/2004:09:20:45 -0800]
 "GET / HTTP/1.1" 200 658
"http://www.example.com/start.html"
"Mozilla/4.08 [en] (Win98; I ;Nav)"
```

También se pueden utilizar múltiples instancias de la directiva `CustomLog` para clasificar diferentes mensajes:

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" common
CustomLog logs/Access_log common
CustomLog logs/referer_log "%{Referer}i -> %U"
CustomLog logs/agent_log "%{User-agent}i"
```

Debido al gran tamaño que pueden alcanzar estos ficheros, se utilizan mecanismos de filtrado.

- `SetEnvl` permite definir una condición sobrelos campos de una petición para posteriormente ser filtrada por la directiva `CustomLog`.

```
SetEnvIf Remote_Addr "127\.0\.0\.1" dontlog
SetEnvIf Request_URI "^/ejemplo\.txt$" dontlog
CustomLog logs/access_log common env=!dontlog
```

En este ejemplo se excluyen aquellos mensajes de log que por producidos tanto por acceso locales como los que pretenden acceder al fichero `ejemplo.txt`.

`SetEnv` admite los siguientes filtros:

Filtro	Significado
Remote_Host	Nombre de cliente
Remote_Addr	IP del cliente
Request_Method	Método usado en la petición (GET, POST…)
Request_Protocol	Versión HTTP
Request_URI	Recurso solicitado (según aparece en la línea de petición)

Se pueden usar expresiones regulares.

Otras directivas relacionadas:

- **`ServerSignature`**: esta directiva permite añadir una línea conteniendo la versión del servidor y el nombre del host a las páginas generadas por el servidor (por ejemplo, páginas de error). Puede valer On, Off o EMail, que incluye una línea mailto: con enlace a `ServerAdmin`.
- **`HostnameLookups`**: esta directiva le indica al servidor web que utilice en los ficheros de log el nombre de los clientes en lugar de su dirección IP, para lo cual deberá buscar el nombre a partir de las direcciones IP en DNS o fichero de host del servidor. No es recomendable activar esta directiva porque cada petición de cliente obliga a realizar búsquedas del nombre, lo que implica un incremento en el tráfico de la red. Por tanto se recomienda dejarlo a off.

```
HostnameLookups Off
```

Se puede realizar la misma funcionalidad de forma diferida mediante el comando `logresolve` que se encuentra en el directorio `bin` de Apache.

5.5.3.5 DIRECTORIOS DE USUARIO

En Unix, cada usuario tiene un directorio asociado considerado su directorio raíz. En Apache se aplica el mismo concepto pero aplicado al **servicio de documentos.**

- **`UserDir`**: en sistemas que tienen múltiples usuarios, cada usuario (o alguno de ellos) puede tener permiso para tener un web site en su home directory usando la directiva `UserDir`. Una URL referenciada a un usuario (como por ejemplo `http://ejemplo.com/~username/`) devolverá el contenido de un directorio en función del valor de la directiva `UserDir` y del nombre del usuario (usualmente el `home_directory` de éste).

  Esta directiva puede tomar distintas formas:

  - Un path que no comienza por slash (/). Se asume que es un path de directorio relativo al `home_directory` del usuario. Por ejemplo:

    ```
    UserDir  public_html
    ```

    Si el servidor recibe la URL `http://ejemplo.com/~username/file.html`, buscará el fichero file `html` en el path

    `/home/username/public_html/`, siendo `/home/username` el `home_directory` de `username`.

  - Un path comenzando por un slash (/). El path del directorio se construirá usando el path indicado en la directiva más el nombre del usuario especificado. Por ejemplo:

    ```
    UserDir  /var/html
    ```

    Si el servidor recibe la URL `http://ejemplo.com/~username/file.html`, buscará el fichero file `html` en el path `/var/html/username/`.

  - Un path que contiene un asterisco (*). El asterisco es reemplazado por el nombre del usuario. Por ejemplo:

    ```
    UserDir  /var/www/*/docs
    ```

    Si el servidor recibe la URL `http://ejemplo.com/~username/file.html`, buscará el fichero file `html` en el path `/var/www/username/docs/`.

  Esta directiva permite también indicar a qué usuarios se les restringe esta funcionalidad. Por ejemplo, las siguientes directivas activan la funcionalidad `UserDir` para todos los usuarios excepto para root, user1, user2 (Nota: desactivarlo para root es siempre altamente recomendable).

```
UserDir enabled
UserDir disabled root user1 user2
```

  Por el contrario, la siguiente desactiva la funcionalidad `UserDir` para todos los usuarios excepto user1 y user2.

```
UserDir disabled
UserDir enabled user1 user2
```

  Esta directiva se debe usar en conjunción con `<Directory>` para limitar las funcionalidades de los directorios de usuario. Por ejemplo:

```
UserDir public_html
<Directory /home/*/public_html>
   AllowOverride FileInfo AuthConfig Limit
```

```
    Options     MultiViews     Indexes     SymLinksIfOwnerMatch
IncludesNoExec
    <Limit GET POST OPTIONS PROPFIND>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS PROPFIND>
        Require all denied
    </LimitExcept>
</Directory>
```

En este caso los directorios de usuario están en el directorio `public_html` de su `home_directory`, y sólo hay acceso en modo lectura.

También se puede hacer que cada usuario tenga su propio directorio `cgi-bin`, con permisos de ejecución de CGIs:

```
UserDir public_html
<Directory /home/*/public_html/cgi-bin/>
    Options ExecCGI
    SetHandler cgi-script
</Directory>
```

Un programa `ejemplo.cgi` puede ser ejecutado desde el directorio `/home/username/public_html/cgi-bin/` con la siguiente URL:

```
http://ejemplo.com/~username/cgi-bin/ejemplo.cgi
```

Si se quiere permitir a los usuarios modificar la configuración del servidor en su espacio web, hay que usar ficheros `.htaccess` en los directorios de usuario y asignar, mediante `AllowOverride`, un valor suficiente a las directivas que se quiere permitir al usuario que modifique.

La notación "~" es específica para Apache.

5.5.3.6 MODIFICACIÓN DEL ESPACIO DE URI

La correspondencia entre URIs y documentos puede ser modificada. Por defecto Apache traduce el sufijo de la URI a un camino en el sistema de ficheros.

- **`Alias`**: esta directiva pone a disposición del servidor documentos existentes fuera del directorio indicado en `DocumenRoot`. Por ejemplo:

```
Alias /icons/path_icons/
```

Cualquier URL que referencia a `/icons/` (por ejemplo, `http://ejemplo.com/icons/escudo.gif`) hará que el sistema busque el fichero en el directorio local `/path_icons/` en lugar del indicado por `DocumenRoot`.

Si se crea un alias a un directorio fuera del `DocumentRoot`, se necesita permitir explícitamente el acceso a dicho directorio. Por ejemplo:

```
Alias /image /ftp/pub/image
<Directory /ftp/pub/image>
    Order allow,deny
    Allow from all
</Directory>
```

- **`AliasMath`** acepta una expresión regular.

5.5.3.7 REDIRECCIÓN DE PETICIONES

A veces la URI recibida debe ser traducida a otra que se sirven un otra URL.

- **Redirect**: la directiva `Redirect` permite mapear una URL vieja en una nueva. La nueva URL es devuelta al cliente, que intenta acceder otra vez con la nueva dirección. Por ejemplo:

```
Redirect /servicio http://otro.servicio:8080/servicio
```

Si el cliente solicita `http://ejemplo.com/servicio/fichero.html`, el servidor le responderá al cliente que acceda a `http://otro.servicio:8080/servicio/fichero.html` en su lugar.

Esta directiva puede tener un primer parámetro que indica el estatus de error que recibirá el cliente. Puede tener uno de los siguientes valores:

- → **Permanent**: indica que el recurso ha sido trasladado permanentemente. Devuelve un estado permanente de redirección (301).
- → **Temp**: es el status por defecto. Devuelve un estado temporal de redirección (302).

Se pueden utilizar otros status numéricos dando como parámetro el número correspondiente.

```
Redirect permanent /servicio http://otro:8080/servicio
Redirect 303 /servicio http://otro:8080/servicio
```

5.5.3.8 OTRAS DIRECTIVAS DEL SERVIDOR PRINCIPAL

- **DirectoryIndex**: esta directiva se aplica para indicar el nombre del fichero o ficheros para usar como índice de directorio, al acceder a URLs de directorio. Puede haber múltiples entradas separadas por espacios. Por ejemplo:

```
DirectoryIndex index.html index.htm index.shtml
```

Una URL del tipo `http://ejemplo.com/docs/` devolverá index.html si existe, o si no `index.htm`, o `index.shtml`. El valor de esta directiva no tiene por qué estar referenciada al directorio. Por ejemplo:

```
DirectoryIndex index.html index.htm /cgi-bin/index.pl
```

Ejecutará el programa `cgi /cgi-bin/index.pl` si en el directorio no existe `index.html` ni `index.htm`.

- **ScriptAlias**: controla qué directorios contienen scripts de servidor. Es esencialmente como un Alias, pero los documentos almacenados en el directorio referenciado son ejecutados como aplicaciones cuando son solicitados. Por ejemplo:

```
ScriptAlias /cgi-bin/"/path/cgi-bin"
```

5.5.4. Directivas de Host Virtual

El término Hosting Virtual se refiere a hacer funcionar más de un sitio web (tales como www.sitio1.com y www.sitio2.com) en una sola máquina y con una sola instancia de Apache. Los sitios web virtuales pueden estar "basados en direcciones IP", lo que significa que cada sitio web tiene una dirección IP diferente, o "basados en nombres diferentes", lo que significa que con una sola dirección IP están funcionando sitios web con diferentes nombres (de dominio). El hecho de que estén funcionando en la misma máquina física pasa completamente desapercibido para el usuario que visita esos sitios web.

Si no se definen servidores virtuales pero un host tiene distintos nombres, cuando se realiza una petición HTTP con cualquiera de dichos nombres será el servidor principal el que

responda la petición. En caso de que cada uno de los nombres tenga un host virtual asociado, una petición realizada con dicho nombre será contestada con el host virtual correspondiente.

Apache fue uno de los primeros servidores web en soportar hosting virtual basado en direcciones IP. Las versiones 1.1 y posteriores de Apache soportan hosting virtual (vhost) basado tanto en direcciones IP como en nombres. Ésta última variante de hosting virtual se llama algunas veces basada en host o hosting virtual no basado en IP.

Los servidores virtuales se definen mediante la directiva contenedor <VirtualHost>. Toda la configuración relacionada con un servidor virtual debe estar comprendida entre el inicio de definición del host (<VirtualHost>) y el fin de definición (</VirtualHost>). Entre ambos tags se puede incluir cualquier directiva de las utilizadas para el servidor principal: cualquier directiva no modificada en el ámbito de definición del host virtual hereda la del servidor principal.

La información en Apache sobre este tema se encuentra en: <httpd.apache.org/docs/2.4/es/vhosts/>

5.5.4.1 DIFERENCIAS ENTRE HOSTING VIRTUAL BASADO EN NOMBRES Y BASADO EN IPS

El hosting virtual basado en IPs usa la dirección IP de la conexión para determinar qué host virtual es el que tiene que servir. Por lo tanto, necesitará tener diferentes direcciones IP para cada host. Si usa hosting virtual basado en nombres, el servidor atiende al nombre de host que especifica el cliente en las cabeceras de HTTP (cabecera Host). Usando esta técnica, una sola dirección IP puede ser compartida por muchos sitios web diferentes.

El hosting virtual basado en nombres es normalmente más sencillo, porque solo necesita configurar su servidor de DNS para que localice la dirección IP correcta y entonces configurar Apache para que reconozca los diferentes nombres de host. Usando hosting virtual basado en nombres también se reduce la demanda de direcciones IP, que en su versión 4 empieza a ser un bien escaso. Por lo tanto, debe usar hosting virtual basado en nombres a no ser que haya alguna razón especial por la cual tenga que elegir usar hosting virtual basado en direcciones IP. Algunas de estas razones pueden ser:

- Algunos clientes antiguos no son compatibles con el hosting virtual basado en nombres. Para que el hosting virtual basado en nombres funcione, el cliente debe enviar la cabecera de host HTTP. Esto es necesario para HTTP/1.1, y está implementado como extensión en casi todos los navegadores actuales.
- Algunos sistemas operativos y algunos elementos de red tienen implementadas técnicas de gestión de ancho de banda que no pueden diferenciar entre hosts a no ser que no estén en diferentes direcciones IP.

5.5.4.2 CÓMO USAR HOSTING VIRTUAL BASADO EN NOMBRES

La base de la definición de hosts virtuales basados en nombre consiste en asociar varios nombres de servidor a un solo host (con una única dirección) y referenciarlos en el método de resolución de nombres que use el usuario (DNS, fichero de hosts). Este método permite

modificar las direcciones IP (de forma directa o dinámica) sin tener que cambiar la configuración de Apache.

Como se ha indicado antes, cualquier directiva de servidor principal puede ser modificada para un host virtual. Las directivas básicas (del módulo core) a utilizar son las que se incluyen en la tabla siguiente:

Módulos relacionados	Directivas relacionadas	Descripción
Core	VirtualHost	Contenedor de las directivas de definición del host virtual
	ServerName	Nombre del host virtual y puerto
	ServerAlias	Permite definir nombres alternativos para el host virtual
	ServerPath	URL para acceder al host virtual con navegadores no compatibles
	DocumentRoot	Describe el directorio raíz del árbol de documentos que va a servir el host virtual

En anteriores versiones, la directiva `NameVirtualHost` se utilizaba para designar la dirección IP de una estructura de hosting virtual basada en nombres, pero ya no tiene ningún efecto a partir de la versión 2.4.

El procedimiento de definición de hosts virtuales basado en nombre es el siguiente:

1. Definir los nombres de hosts y la dirección IP a utilizar. Crear los registros correspondientes (en los ficheros de hosts o en DNS) para que los nombres sean correctamente traducidos.

```
www.sitio1.com  10.23.21.5
www.sitio2.com  10.23.21.5
www.sitio3.com  10.23.21.5
```

2. El siguiente paso es crear un bloque <`VirtualHost`> para cada host diferente que quiera alojar en el servidor. El argumento de la directiva <`VirtualHost`> indica qué dirección IP, y posiblemente puerto, que se va a usar para atender las peticiones de dicho Host (por ejemplo, una dirección IP, o un * para usar todas las direcciones que tenga el servidor). Dentro de cada bloque <`VirtualHost`>, necesitará como mínimo una directiva `ServerName` para designar qué host se sirve y una directiva `DocumentRoot` para indicar dónde están los contenidos a servir dentro del sistema de ficheros.

En el siguiente ejemplo, los hosts virtuales definidos serán accedidos respectivamente mediante las URLs `http://www.site1.com`, `http://www.site2.com` y `http://www.site3.com`, respectiva-mente.

```
<VirtualHost 10.23.21.5:80>
    ServerName www.site1.com
    ServerAlias site1.com *.site1.com
```

```
    DocumentRoot /www/site1
</VirtualHost>

<VirtualHost 10.23.21.5:80>
    ServerName www.site2.com
    DocumentRoot /www/site2
</VirtualHost>

<VirtualHost 10.23.21.5:80>
    ServerName www.site3.com
    DocumentRoot /www/site3
</VirtualHost>
```

Para acceder a un determinado sitio web usando diferentes nombres, se utiliza la directiva `ServerAlias`. En el ejemplo, en el primer bloque <`VirtualHost`> de arriba, la directiva `ServerAlias` indica la lista de nombres que pueden usarse para acceder al mismo sitio web.

Los permisos de acceso a cada uno de los directorios especificados como `DocumentRoot` deberán ser establecidos mediante las correspondientes directivas de control de acceso de directorio.


Aunque se especifique una dirección IP como argumento, esto no hace que el servidor escuche automáticamente en esa dirección IP. Además cualquier dirección IP especificada debe asociarse con un dispositivo de red del servidor.

5.5.4.3 AÑADIR HOSTS VIRTUALES A UN SERVIDOR WEB YA EXISTENTE

Si está añadiendo hosts virtuales a un servidor web ya existente, deberá crear también un bloque <`VirtualHost`> para el host que ya tenga funcionando (el servidor principal). Los valores de las directivas `ServerName` y `DocumentRoot` desde este nuevo host virtual deben tener los mismos valores que los de las directivas `ServerName` `DocumentRoot` globales. Ponga este host virtual como el primero en el archivo de configuración para que sea el que actúe como host por defecto.

Por ejemplo, suponga que está sirviendo el dominio www.sitio.com y quiere añadir el host virtual www.sitio1.com, que apunta a la misma dirección IP. Entonces, lo único que tiene que hacer es añadir lo siguiente al fichero `httpd.conf`:

```
ServerName www.sitio.com

<VirtualHost *:80>
    ServerName www.sitio.com
    ServerAlias sitio.com *.sitio.com
    DocumentRoot /www/htdocs
</VirtualHost>

<VirtualHost *:80>
    ServerName www.sitio1.com
    DocumentRoot /www/sitio1
</VirtualHost>
```

En el ejemplo, las peticiones para todos los hosts en el dominio sitio.com serán servidas por el host virtual www.sitio.com. Los carácteres comodines * y ? pueden usarse para encontrar equivalencias con los nombres.


No puede inventarse nombres y ponerlos en la directiva `ServerName` o `ServerAlias`. Primero debe tener su servidor de DNS debidamente configurado para que pueda hacer corresponder esos nombres con una dirección IP de su servidor.

Para terminar, puede mejorar el rendimiento de la configuración de los hosts virtuales poniendo otras directivas dentro de las secciones <`VirtualHost`>. La mayor parte de las directivas pueden ponerse en esos contenedores y cambiarán solo la configuración del host virtual al que se refieran. Para ver si una directiva en particular puede usarse así, consulte el Contexto de la directiva. Las directivas de configuración especificadas en el contexto del servidor principal (fuera de cualquier sección <`VirtualHost`>) se usan única y exclusivamente si sus valores no son sustituidos por alguno de los parámetros de configuración del host virtual.

Cuando llega una petición, el servidor primero verifica si se está usando una dirección IP que coincide con el valor de la directiva `NameVirtualHost`. Si es el caso, mirará en cada sección <`VirtualHost`> cuya IP coincida e intentará encontrar si el valor de la directiva `ServerName` o de la directiva `ServerAlias` coincide con el nombre del sitio web de la petición. Si encuentra una coincidencia, usa la configuración de ese servidor. Si no la encuentra, usa el primer host virtual de la lista cuya dirección IP coincida con el de la petición.

Como consecuencia, el primer host virtual de la lista es el que se usa por defecto. La directiva `DocumentRoot` del servidor principal no se usará nunca cuando una dirección IP coincida con el valor de la directiva `NameVirtualHost`. Si quiere usar una configuración especial para peticiones que no coinciden con ningún host virtual en concreto, ponga esa configuración en una sección <`VirtualHost`> y póngala la primera en el fichero de configuración.

La documentación de Apache sobre este tema se encuentra en:
<httpd.apache.org/docs/2.4/es/vhosts/name-based.html>

5.5.4.4 CÓMO USAR HOSTING VIRTUAL BASADO EN IPS

Para poder utilizarlo, la máquina ha de tener varias direcciones IP. Esto puede conseguirse, poniéndole varias tarjetas de red, o configurando la red y el sistema operativo con interfaces virtuales, del tipo “ip aliases”, con el comando `ifconfig`.

El hosting virtual basado en IPs es mucho más raro de ver y más complejo de configurar. Es menos flexible que el basado en nombres puesto que Apache debe ser configurado incluyendo las direcciones IP de forma estática.

Las directivas básicas empleadas son las siguientes:

Módulos relacionados	Directivas relacionadas	Descripción
Core	VirtualHost	Contenedor de las directivas de definición del host virtual
	ServerName	Nombre del host virtual y

		puerto
	ServerAdmin	Incluye la dirección de contacto del administrador del servidor
	ServerPath	URL para acceder al host virtual con navegador no compatibles
	ErrorLog	Ubicación de registro de errores
	TrasferLog / CustomLog	Registro de las peticiones al servidor
	DocumentRoot	Describe el directorio raíz del árbol de documento que va a servir el host virtual

El procedimiento a seguir es el siguiente:

1. Definir los nombres de hosts a utilizar y las direcciones IP asociadas. Incluir la asociación en el método de referenciación usado por los usuarios (DNS, ficheros de hosts).

```
www.sitio1.com   10.23.21.4
www.sitio1.com   10.23.21.5
www.sitio1.com   10.23.21.6
```

2. Definir los hosts virtuales para cada una de las direcciones IP

```
<VirtualHost 10.23.21.4:80>
  ServerAdmin webmaster@www.site1.com
  ServerName www.site1.com
  ServerAlias site1.com *.site1.com
  DocumentRoot /www/site1
  ErrorLog  /www/logs/site1/error_log
  CustomLog  /www/logs/site1/access_log combined
</VirtualHost>

<VirtualHost 10.23.21.5:80>
  ServerAdmin webmaster@www.site2.com
  ServerName www.site2.com
  DocumentRoot /www/site2
  ErrorLog  /www/logs/site2/error_log
  CustomLog  /www/logs/site2/access_log combined
</VirtualHost>

<VirtualHost 10.23.21.6:80>
  ServerAdmin webmaster@www.site2.com
  ServerName www.site3.com
  DocumentRoot /www/site3
  ErrorLog  /www/logs/site3/error_log
  CustomLog  /www/logs/site3/access_log combined
</VirtualHost>
```

La documentación de Apache sobre este tema se encuentra en:
<httpd.apache.org/docs/2.4/es/vhosts/ip-based.html>

5.1. Códigos de estado HTTP más comunes

Se puede ver el resultado de la petición HTTP, con los códigos de estado (Status Code) especificados en el RFC 2616. El código está formado por 3 dígitos, el primero de ellos indica la clase de respuesta.

- 1xx: Petición recibida, continuando el proceso.
- 2xx: Són códigos de éxito. La petición fue correctamente recibida y aceptada.
- 3xx: Códigos de redirección. Se requieren llevar a cabo más acciones para completar la petición.
- 4xx: Error por parte del cliente.
- 5xx: Error por parte del servidor.

A continuación los veremos con mayor detalle:

5.1.1. Códigos de éxito

Código	Descripción
200	OK. Respuesta estándar para peticiones correctas.
201	Creado. La petición ha sido aceptada y se ha creado un nuevo recurso.
202	Aceptada. La petición ha sido aceptada para ser procesada, pero el procesamiento no ha sido completado aún.
204	Sin contenido.
206	Contenido parcial. La petición servirá parcialmente el contenido solicitado. Esta característica es utilizada por herramientas de descarga como "wget" para continuar la trasferencia de descargas anteriormente interrumpidas, o para dividir una descarga y procesar las partes simultáneamente.

5.1.2. Códigos de redirección

Código	Descripción
300	Múltiples opciones. Indica opciones múltiples para el URI que el cliente podría seguir. Esto podría ser utilizado por ejemplo, para presentar distintas opciones de formato para video o listar archivos con distintas extensiones.
301	Movido permanentemente. Esta y todas las peticiones futuras deberán ser dirigidas a la URI dada.
302	Movido temporalmente. Este es el código de redirección más popular.
304	No modificado. Indica que la petición a la URL no ha sido modificada desde que fuera requerida por última vez. Típicamente, el cliente HTTP provee un encabezado como *If-Modified-Since* para indicar una fecha y hora contra la cual el servidor pueda comparar. El uso de este encabezado ahorra ancho de banda y reprocesamiento tanto del servidor como del cliente.

305	Utilice un proxy. Muchos clientes HTTP no se apegan al estándar al procesar respuestas con este código, principalmente por motivos de seguridad.
307	Redirección temporal. Se trata de una redirección que debería haber sido hecha con otra URI, sin embargo aún puede ser procesada con la URI proporcionada. En contraste con el código 303, el método de la petición no debería ser cambiado cuando el cliente repita la solicitud. Por ejemplo, una solicitud POST tiene que ser repetida utilizando otra petición POST.

5.1.3. Códigos de error de cliente

Código	Descripción
400	Solicitud incorrecta. La solicitud contiene sintaxis errónea y no debería repetirse.
401	No autorizado. Similar al 403 Forbidden, pero específicamente para su uso cuando la autentificación es posible pero ha fallado o aún no ha sido provista.
402	Pago requerido. La intención original era que este código pudiese ser usado como parte de alguna forma o esquema de dinero electrónico, pero eso no llegó a producirse y esté código nunca se utilizó.
403	Prohibido. La solicitud fue legal, pero el servidor se rehúsa a responderla.
404	No encontrado. Recurso no encontrado. Se utiliza cuando el servidor web no encuentra la página o recurso solicitado.
405	Método no permitido. Una petición fue hecha a una URI usando un método de solicitud no soportado por dicha URI. Por ejemplo, cuando se usa GET en una forma que requiere que los datos sean enviados vía POST.
406	No aceptable. El servidor no es capaz de devolver los datos en ninguno de los formatos aceptados por el cliente, indicados por éste en la cabecera *Accept* de la petición.
407	Autenticación Proxy requerida.
408	Tiempo de espera agotado. El cliente falló al continuar la petición.
409	Conflicto. Indica que la solicitud no pudo ser procesada debido a un conflicto con el estado actual del recurso.

5.1.4. Códigos de error del servidor

Código	Descripción
500	Error interno. Es un código comúnmente emitido por aplicaciones empotradas en servidores web, las mismas que generas contenido dinámicamente.
501	No implementado.
502	Gateway incorrecto.
503	Servicio no disponible.
504	Tiempo de espera de gateway agotado.
505	Versión de HTTP no soportada.

5.2. Conclusiones

- **Apache no dispone de herramientas de administración**, aunque hay algunas en el mercado que pueden facilitar la tarea. Algunos sistemas operativos, como Red Hat, tienen instaladas herramientas de configuración.
- **Apache se administra mediante la edición y modificación de las directivas en el fichero httpd.conf.** Puede haber otros ficheros que se cargan en el principal mediante la directiva `include`. También se puede modificar la configuración a nivel de directorio mediante la creación de ficheros `.htaccess`.
- Hay tres grandes **grupos de directivas**: las que afectan a toda la instancia (parámetros **globales**), las que afectan al servidor que escucha y responde por defecto (directivas del **servidor principal**) y las de los **servidores virtuales**.
- Entre los **parámetros globales** se incluye la carga de los módulos dinámicos mediante el **soporte DSO** y la directiva **`LoadModule`**. Una directiva relacionada con un módulo no funcionará e incluso puede provocar errores si el módulo correspondiente no ha sido cargado.
- Los módulos **MPM no pueden ser cargados dinámicamente**, sino que se instalan en tiempo de compilación. Es conveniente usar el módulo por defecto para el sistema operativo de que se trate, porque si se usa uno no admitido por el Sistema Operativo Apache no podrá funcionar.
- Hay que tener cuidado al crear hosts virtuales basados en nombres en una instalación ya existente porque es necesario redefinir el servidor principal como si fuese un host virtual.
- Es importante conocer los códigos de error que devuelve el sistema (se usan en las cabeceras HTTP de respuesta y en los ficheros de `log` y `access`) para poder hacer un seguimiento en caso de problemas.