INAP
INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

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

Módulo 2. Apache Web Server

Capítulo 7. Otros Módulos de Apache Web Server

Autores

Janine García Morera

Alexandra López de la Oliva Portugués

Julio Villena Román

Octubre de 2014

Índice de contenidos

CAPÍTULO 7 OTROS MÓDULOS DE APACHE WEB SERVER

7.1. Introducción

Hasta ahora hemos visto algunos de los módulos básicos que permiten a Apache funcionar como un servidor Web estándar, con gran potencia y fiabilidad.

Sin embargo, **una de las grandes potencialidades de Apache es la de poder incrementar sus posibilidades mediante la carga de nuevos módulos**, que pueden ser módulos propios o módulos de terceros. Algunos fueron módulos de tercero en su día y han sido incorporados en la distribución estándar de Apache.

Estos módulos pueden haber sido cargados estáticamente en tiempo de compilación o se pueden cargar de forma dinámica al levantar la instancia.

Los módulos que se verán permiten a Apache actuar como un servidor Proxy, aumentar la visibilidad que un usuario tiene del servidor Web que se le presenta con el "mapeo" de otros servidores Web (que pueden o no ser servidores Apache), configurar una granja de servidores Web con un servidor Apache que actúa como frontal de balanceo y securizar todo el entorno haciendo que este servidor se comunique con el usuario mediante SSL.

7.2. Objetivos

- Conocer la posibilidad de usar nuevos módulos no activados en la configuración por defecto.
- Conocer una pequeña parte de las posibilidades de ampliación de funcionalidades que Apache nos ofrece.
- Conocer la forma de aumentar el rendimiento de nuestra implementación por medio de la puesta en marcha de nuevos servidores.
- Aprender a securizar nuestra instalación usando para ello el soporte SSL que suministra el módulo `mod_ssl`.

7.3. Soporte Proxy

El soporte Proxy es ofrecido por los módulos `mod_proxy` y los módulos correspondientes a los distintos protocolos: `mod_proxy_http`, `mod_proxy_ftp`, `mod_proxy_ajp` y `mod_proxy_connect`. Para activar el soporte Proxy es necesario cargar los módulos correspondientes:

- `LoadModule proxy_module modules/mod_proxy.so`
- `LoadModule proxy_ajp_module modules/mod_proxy_ajp.so`

- `LoadModule proxy_connect_module modules/mod_proxy_connect.so`
- `LoadModule proxy_http_module modules/mod_proxy_http.so`

Apache puede actuar como proxy de dos tipos: **forward proxy** y **proxy inverso**.

- Un **fordward proxy**, o proxy, es un servidor que se sitúa entre el cliente y el servidor y que permite al cliente obtener contenido del servidor enviando la petición a éste a través del proxy, que remite la petición al servidor y devuelve la respuesta al cliente. El cliente (normalmente un navegador) debe estar configurado explícitamente para usar el Proxy como elemento intermedio para acceder al servidor.

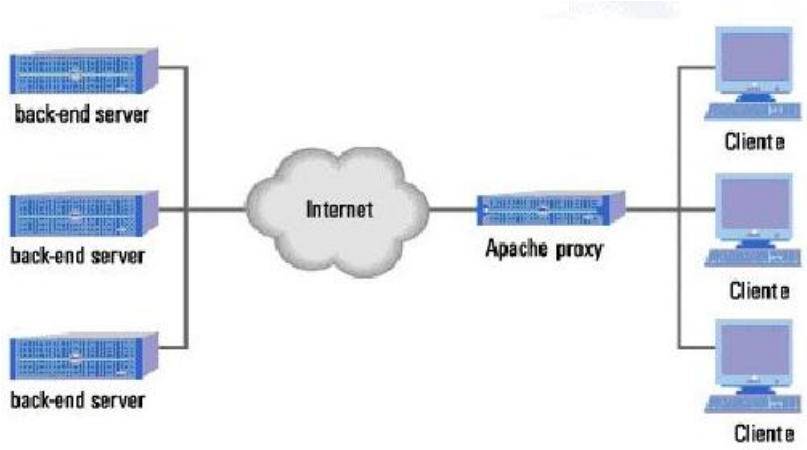

Figura 2.7.1: Fordward Proxy

El forward proxy se activa mediante la directiva `ProxyRequests`:

```
ProxyRequests On
ProxyVia On
<Proxy *>
    Require host internal.example.com
</Proxy>
```

- Un **Proxy inverso** aparece ante el cliente como un servidor web, sin que el cliente necesite realizar ningún tipo de configuración especial. El cliente hace peticiones en el área de direccionamiento de este web server proxy, que decide dónde tiene que remitir la petición y devuelve el contenido como si el mismo fuese el origen.

  Situado enfrente de otros servidores, un reverse proxy provee un frontal unificado a otros Web server de back-end. Por ejemplo, si un cliente solicita la URL `http://ejemplo.com`, la petición la recibe un servidor reverse proxy que sirve la petición al cliente. El reverse proxy contacta con uno de los servidores de back-end para servir la petición del cliente. El cliente se comunica sólo con el servidor de reverse proxy.

  Un Proxy inverso se activa mediante las directivas `ProxyPass` y `ProxyPassReverse`.

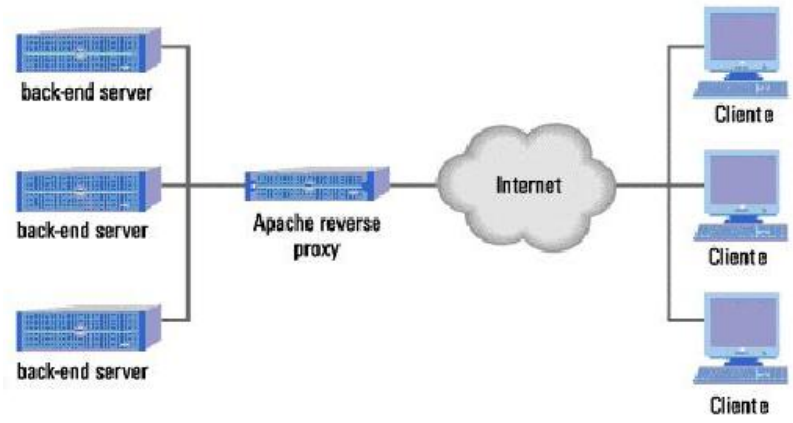

Figura 2.7.2: Proxy Inverso

```
ProxyRequests Off

<Proxy *>
    Require all granted
</Proxy>

ProxyPass /foo http://foo.example.com/bar
ProxyPassReverse /foo http://foo.example.com/bar
```

Al acceder a la URL `/foo`, internamente el servidor nos presenta `foo.example.com/bar`.

Conceptos importantes
El soporte Proxy es ofrecido por los módulos `mod_proxy` y los módulos correspondientes a los distintos protocolos: `mod_proxy_http`, `mod_proxy_ftp`, `mod_proxy_ajp` y `mod_proxy_connect`. Para activar el soporte Proxy es necesario cargar los modulos correspondientes.
Apache puede actuar como proxy de dos tipos: forward proxy y proxy inverso.

7.4. Soporte de Balanceo de Carga

A partir de la versión 2.1 de Apache se incluye un módulo de balanceo de carga a través de **`mod_proxy`**, llamado **`mod_proxy_balancer`**.

El módulo `mod_proxy_balancer` suministra soporte de balanceo de carga para los protocolos HTTP, FTP y AJP1.3. Requiere el servicio del módulo `mod_proxy`. Para que se pueda ofrecer la posibilidad de balanceo de carga, los módulos `mod_proxy` y

`mod_proxy_balancer` deben estar presentes en el servidor, cargados bien de forma estática (en compilación), bien de forma dinámica (mediante la directiva `LoadModule`).

Hay dos algoritmos disponibles de balanceo de carga: por número de solicitudes servidas y por ocupación de ancho de banda. Ambos se controlan a través de los módulos **`mod_lbmethod_byrequests`** y **`mod_lbmethod_bytraffic`** respectivamente.

- **Algoritmo por número de solicitudes**: Se activa mediante el módulo `mod_lbmethod_byrequests` con el parámetro `lbmethod=byrequests`. La idea es que el distribuidor (scheduler) reparte las peticiones entre varios workers (instancias de otros servidores remotos, que pueden ser servidores web, servidores FTP o servidores de aplicaciones) en función del valor del parámetro `lbfactor` (que representa cuanto se espera que trabaje un worker, es decir, el peso que tiene en el reparto de trabajo).
- **Algoritmo por ocupación por ancho de banda:** Se activa mediante el módulo `mod_lbmethod_bytraffic` con el parámetro `lbmethod=bytraffic`. El parámetro `lbfactor` indica cuánto tráfico, en bytes, queremos que maneje este worker. Realmente representa el peso de la cantidad de trabajo a realizar, pero en lugar de contar el número de peticiones que sirve se cuenta la cantidad de tráfico (en bytes) que procesa el worker.
- **Algoritmo por ocupación de peticiones:** Se activa mediante el módulo `lbmethod=bybusyness`. En esta opción el distribuidos (scheduler) mantiene un registro de cuántas peticiones se encuentran actualmente atendidas por casa servidor. La petición se asigna automáticamente a aquél con el menor número de solicitudes activas. (Útil para reducir la latencia). En caso de empate se utilizará el factor para romperlo. Con el tiempo, la distribución del trabajo llega a parecerse al algoritmo byrequest.

La forma de implementar balanceo de carga mediante Proxy es la siguiente:

```
<Proxy balancer://misistema>
    BalancerMember http://servidor1
    BalancerMember http://servidor2
    BalancerMember http://servidor3
</Proxy>
ProxyPass /balanceo balancer://misistema/
```

Por defecto se utiliza el balanceo (byrequest).

Esta configuración balancea los accesos a la URL `/balanceo` entre los servidores servidor1, servidor3 y servidor3.

7.4.1. Activación del soporte del controlador del balanceador

Este módulo requiere del servicio del módulo `mod_status`. El controlador del balanceador (balancer manager) posibilita la modificación dinámica de los miembros de balanceo, pudiéndose modificar el factor de balanceo de un miembro en particular o ponerlo en modo off-line.

Los módulos `mod_status` y `mod_proxy_balancer` tienen que estar presentes en el servidor, bien de forma estática o bien de forma dinámica.

Se puede limitar el acceso al controlador del balanceador mediante el uso de las directivas de control de acceso: en el ejemplo siguiente se puede acceder al controlador de balanceo desde un navegador accediendo a la página `http://miservidor/balancer-manager` desde un puesto del dominio `map.es`.

```
<Location /balancer-manager>
    SetHandler balancer-manager
    Require host .map.es
</Location>
```

Usando este controlador se pueden modificar de forma dinámica los parámetros del balanceador:

Load Balancer Manager for localhost

Server Version: Apache/2.4.10 (Win64) PHP/5.5.15
Server Built: Jul 17 2014 12:58:29
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://micluster [p7902e3fa_micluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	(None)	On	0	2	byrequests	/hosts	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://host1			1	0	Init Ok	11	0	-2	6.5K	2.4K
http://host2			1	0	Init Ok	10	0	1	5.9K	2.1K
http://host3			1	0	Init Ok	10	0	1	5.9K	2.1K

Figura 2.7.3: Información generada por balancer-manager

También se pueden modificar los parámetros de cada uno de los miembros, haciendo “clic” encima del worker, por ejemplo para host1 del ejemplo:

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://host1			1	0	Init Ok	11	0	-2	6.5K	2.4K
http://host2			1	0	Init Ok	10	0	1	5.9K	2.1K
http://host3			1	0	Init Ok	10	0	1	5.9K	2.1K

Edit worker settings for http://host1

Load factor:	1			
LB Set:	0			
Route:				
Route Redirect:				
Status:	Ignore Errors	Draining Mode	Disabled	Hot Standby
	On ○ Off ◉	On ○ Off ◉	On ○ Off ◉	On ○ Off ◉
Submit				

Figura 2.7.4: Información generada por balancer-manager, editando un host

Conceptos importantes

El módulo `mod_proxy_balancer` suministra soporte de balanceo de carga para los protocolos HTTP, FTP y AJP1.3. Requiere el servicio del módulo `mod_proxy`. Para que se pueda ofrecer la posibilidad de balanceo de carga, los módulos `mod_proxy` y `mod_proxy_balancer` deben estar presentes en el servidor.

7.5. Soporte SSL

Uno de los aspectos más importantes en las aplicaciones de hoy en día, sobre todo cuando el acceso se realiza a través de Internet, es el de la seguridad. Dotar a las aplicaciones del mayor nivel de seguridad posible es primordial, y uno de los puntos más importantes es dotar al usuario de un canal seguro de comunicación con el servidor. Una de las formas de conseguirlo es utilizando el protocolo SSL.

El protocolo SSL (Secure Sockets Layer) proporciona un canal seguro (encriptado) entre el cliente (navegador) y el servidor.

SSL se puede utilizar tanto en el servidor Web (Apache) como en el contenedor de servlets (Tomcat). Como mínimo es aconsejable habilitar SSL en Apache, puesto que todas las conexiones entre el usuario y la aplicación pasan por él, pero si el servidor Tomcat no reside en la misma máquina que el servidor Apache es aconsejable también habilitar SSL en el servidor Tomcat, lo que permitiría encriptar las conexiones entre Apache y Tomcat.

El protocolo SSL se basa en el uso de técnicas de encriptación de clave pública, o criptografía asimétrica, cuyo soporte básico es la utilización de algoritmos de encriptación que usan dos claves, de tal manera que si un mensaje se encripta con una clave debe ser desencriptado con la otra. De esta forma, una de las claves se hace pública (la clave pública) y la otra permanece oculta (la clave privada), de manera que cualquiera puede enviar un mensaje cifrado con la clave pública del receptor, pero sólo éste puede descifrarlo puesto que sólo él tiene la clave privada.

Figura 2.7.5: Clave pública y privada

El hecho de que el mensaje se reciba cifrado no evita la posibilidad de que el mensaje original haya sido modificado antes de ser cifrado por alguien distinto al usuario. Una forma de asegurar que el mensaje original no ha sido modificado es crear un pequeño resumen del mensaje y una información añadida (como un número secuencial), de tamaño fijo, llamado hash o message digest, cifrarlo con la clave privada del emisor (conformando lo que se llama Firma Digital) y enviarlo junto con el mensaje (que puede estar o no cifrado con la clave pública del receptor). El receptor toma el mensaje, lo descifra con su clave privada (si está cifrado), toma el hash y lo descifra con la clave pública del emisor. Al mismo tiempo, el receptor genera su propio resumen (hash) del mensaje original y lo compara con el descifrado que venía anexado con el mensaje: si son iguales, el mensaje original no ha sido modificado y además se confirma que lo ha remitido quien dice haberlo hecho (principio de no repudio).

La seguridad de que el par de claves pública-privada del comunicante es de éste y no de un tercero se consigue mediante el uso de certificados y de un ente en el que se confía llamado Autoridad de Certificación: se define Certificado como “Una información que se almacena para autentificar entidades de red tales como un servidor o un cliente, conteniendo pieza de información sobre su poseedor (sujeto) y sobre la Autoridad de Certificación que lo firma, más la clave pública del propietario y la firma de la Autoridad de Certificación”. Una Autoridad de Certificación es una “entidad externa de confianza cuyo fin es firmar certificados para las

entidades de red que previamente ha autentificado usando medios seguros, normalmente presenciales. Otras entidades de red pueden solicitar a la Autoridad de Certificación la verificación de la identidad del poseedor del certificado mediante la validación de éste".

Un certificado asocia una clave pública con la identidad real de un individuo, servidor u otra entidad, conocida como un sujeto. Esta información incluye la identificación (DN o Distinguished Name), la clave pública, la identificación y firma de la Autoridad de Certificación que firmó y expedió el certificado y el periodo de tiempo durante el cual el certificado es válido. Puede contener también información adicional (extensiones) así como información administrativa para la Autoridad de Certificación, como puede ser un número de serie.

La información básica de un certificado se resume en el siguiente esquema:

Sujeto	DN, clave pública
Emisor	DN, firma
Periodo de validez	Fecha de inicio, fecha de caducidad
Información Administrativa	Versión, nº de serie
Extensiones	Limitaciones, flags

El DN (Distinguished Name) definido por el estándar X509 (actualmente en su versión 3) contiene los siguientes campos:

Campo DN	Abrev	Descripción	Ejemplo
Common Name	CN	Nombre del propietario	CN=José López
Organización o empresa	O	Nombre de la organización	O=MAP
Unidad Organizacional	OU	Unidad en el organismo o empresa	OU=INAP
Ciudad / Localidad	L	Nombre de la localidad	L=MADRID
Estado / Provincia	ST	Nombre del estado o provincia	ST=MADRID
País	C	Nombre del país (ISO)	C=SP

La obligatoriedad o no de cada campo es potestad de cada Autoridad de Certificación.

El certificado se almacena en formato binario, siguiendo las reglas definidas por la notación ASN.1. Para transmisiones que no pueden manipular formatos binarios, el binario se traduce a ASCII usando una codificación en Base64. Cuando este código se ubica entre delimitadores Begin y End Certificate, esta versión de código es llamada Certificado en codificación PEM ("Privacy Enhanced Mail"). Un ejemplo de este tipo de codificación es la siguiente (fichero `servidor.crt`):

```
-----BEGIN CERTIFICATE-----
MIICkTCCAfoCCQCMYDjMP3fMHDANBgkqhkiG9w0BAQQFADCBjDELMAkGA1UEBhMC
U1AxCzAJBgNVBAgTAk1EMQ0wCwYDVQQHEwRJTkFQMRAwDgYDVQQKEwdBbnRvbmlv
```

```
MQwwCgYDVQQLEwNNQVAxGjAYBgNVBAMTEW1pcG9ydGF0aWwubWFwLmVzMSUwIwYJ
KoZIhvcNAQkBFhZhbnRvbmlvLnBpcXVlcm9AbWFwLmVzMB4XDTA2MDUxMzA3MjEw
MVoXDTA3MDUxMzA3MjEwMVowgYwxCzAJBgNVBAYTAlNQMQswCQYDVQQIEwJNRDEN
MAsGA1UEBxMESU5BUDEQMA4GA1UEChMHQW50b25pbzEMMAoGA1UECxMDTUFQMRow
GAYDVQQDExFtaXBvcnRhdGlsLm1hcC5lczElMCMGCSqGSIb3DQEJARYWYW50b25p
by5waXF1ZXJvQG1hcC5lczCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAo1sz
G+uFqfRKxiocPhUWP1H+AF3xuvqovQC7V0qKmK8rC44tJxtiKNWzHIzSkV6FZk9z
MAN+iN8Ey7HTsDIwoAzkI1ERtko+LIIgg9Ev4tsFL738A6Ko/9cpV5feUA/WmK+J
l/dWs8aHuWgB5xJRlDt0ymtv60f2dtFGu7ZMOQsCAwEAATANBgkqhkiG9w0BAQQF
AAOBgQAlxS0T0Tqyt01PFZ0P7olS9j4Fe8jSk2k71X8waeAXQBwwQnRkMjnVy1Ej
y8NqQnfdnVPuR12zqzgkRuWwYbgOAFH0mhe1N//wKBWv69ExYqGbz1/tHg8ZTUTk
RcxMQYcxLzObKNrU8drwF6hOkFDOFlBOLaDQ2hPp872zMlGwew==
-----END CERTIFICATE-----
```

7.5.1. Análisis de una conexión SSL

El proceso de una conexión SSL entre un navegador y un servidor securizado es el siguiente:

1. Al teclear una URL SSL, el navegador envía un mensaje de saludo al servidor.
2. El servidor envía su propio Certificado y una secuencia aleatoria encriptada con su clave pública.
3. El navegador recibe el certificado del servidor, lo verifica, toma la clave pública del certificado y autentifica el servidor.
4. El cliente genera la clave maestra y la encripta con la clave pública del servidor. La envía, en su caso, junto con su propio Certificado al servidor, y con una secuencia aleatoria encriptada con su clave pública.
5. El servidor recibe el certificado del cliente y lo verifica. Toma la clave pública del certificado y autentifica al cliente, descifra con su clave privada la clave maestra generada por el cliente que pasa a ser la clave secreta para las sucesivas transacciones entre el cliente y el servidor.

Realmente la clave maestra es una suite completa de cifrado en la que se negocia el método de intercambio de clave, el tipo de cifrado y el tipo de función hash.

7.5.2. Configuración de SSL en Apache sobre Windows

La configuración SSL de Apache consiste en generar una petición de certificado, instalar el certificado, habilitar Apache para que escuche por el puerto 443 (el estándar SSL) y gestione la conexión con el cliente. Hay varias soluciones posibles para dotar a Apache de soporte SSL: Apache-SSL (opción gratuita disponible en <http://www.apache-ssl.org/>), el módulo mod_ssl, que se distribuye con Apache a partir de la versión 2.0, y que proviene de Apache-SSL, el comercial Secure Web Server de Red Hat, o el también comercial Módulo SSL Raven de Covalent. En nuestra instalación vamos a configurar el módulo `mod_ssl` para dotar a Apache de soporte SSL.

- **Primer paso**: Configurar el fichero httpd.conf.

Se trata de configurar Apache para que escuche en el puerto SSL (443) además de en el puerto estándar HTTP (el 80). Para ello se utilizará la directiva `Listen`, añadiendo a la entrada ya existente (`Listen 80`) otra con el nuevo puerto:

```
Listen 80
Listen 443
```

Es importante asegurarse de que la directiva `ServerName` está correctamente configurada, ya que es un dato que se incluirá en el certificado.

```
ServerName miservidor.com
```

Para comprobar que todo está correcto, se reinicia Apache y se comprueba que el servidor escucha en ambos puertos tecleando en el navegador las URLs correspondientes: `http://miservidor.com` y `http://miservidor.com:443`. En ambos casos la respuesta debe ser la misma.

- **Segundo paso**: Obtener Apache con soporte SSL

  Se puede obtener una versión de Apache compatible con SSL compilando el código fuente con la opción de configuración --enable-ssl o bien descargar una versión binaria que lo sea. En el caso de sistemas operativos Windows es más sencillo obtener la versión binaria que disponer de las herramientas necesarias para realizar la compilación. En caso de optar por compilar el código fuente, es necesario instalar OpenSSL (<www.openssl.org>).

  Desde Windows: Si se siguió la Instalación de la última versión de Apache en Windows con ApacheLounge. OpenSSL viene ya incluído con el paquete binario de Apache. Por lo que no hace falta descargar la versión binario aparte.

  NOTA: Si fuera necesario descargarlo se pueden obtener en: (<https://www.openssl.org/source/>)

- **Tercer paso**: Obtener e instalar el certificado de prueba. En este ejemplo supondremos que lo obtenemos desde Windows.

  Antes de comenzar este procedimiento es necesario crear el siguiente directorio: `C:\openssl-1.0.1j-win64\ssl` y copiar dentro el fichero `openssl.cnf` que encontrarás en `C:\Apache24\conf`.

  A continuación, seguir el siguiente procedimiento:

  1. Generamos un certificado para crear la clave del servidor.

     ```
     Openssl genrsa –des3 –out server.key 1024
     ```

     Una vez tecleada te preguntara por un “pass phrase”. Escríbela y repítela a continuación. Es importante almacenar tanto la `server.key` como la pass-phrase en una localización segura.

  2. Desencriptamos la llave anterior con el siguiente comando.

     ```
     Openssl rsa –in server.key –out server.pem
     ```

  3. A continuación creamos una solicitud de firma de certificado SSL (CSR) en el fichero `server.csr` y una clave privada (en el fichero `keyfile.pem`).

     ```
     Openssl req –new –key server.key –out server.csr
     ```

     Te pedirá la “pass phrase” y a continuación comenzará a preguntar para incorporar la información necesaria al certificado.

4. Finalmente, obtener un certificado con firma propia, bien solicitándolo a una Autoridad de Certificación (para lo cual habrá que enviar el fichero de solicitud `servidor.csr`) o bien generando un certificado con firma propia (para entornos de prueba) mediante el siguiente comando

```
Openssl x509 -req -days 30 -in server.csr -signkey
server.key -out server.crt
```

El certificado generado mediante este comando tendrá una validez de 30 días

A continuación se muestra una captura de pantalla con todo el procedimiento:

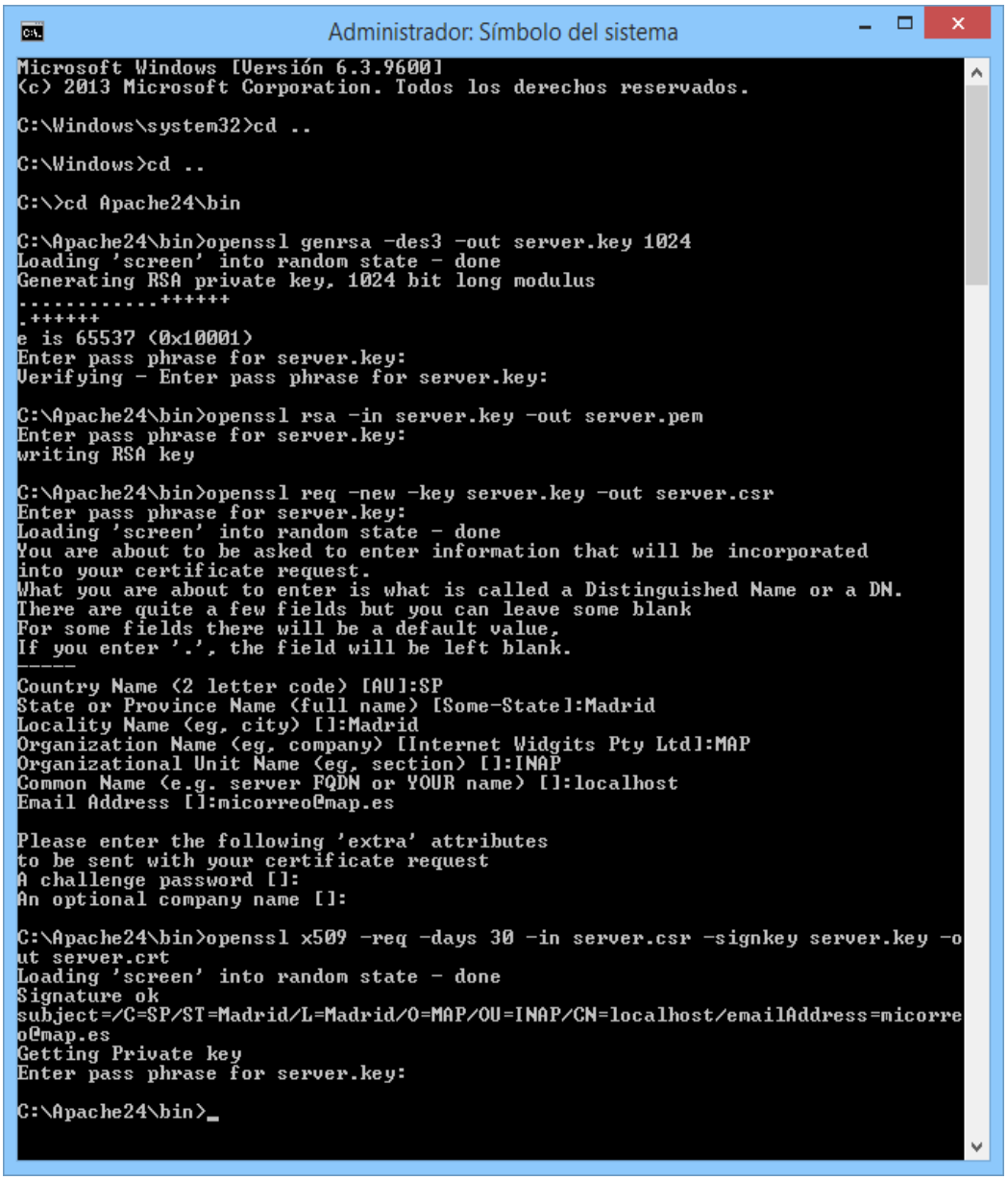

Figura 2.7.6: Pasos para realizar la configuración

Guarda una copia de `server.crt` y `server.pem` en el directorio: `C:/Apache24/conf`.

- **Cuarto paso**: Configurar `mod_ssl`.

Para ello hay que seguir el siguiente método:

1. Comprueba que se encuentra el módulo `mod_ssl.so`.
2. Editar el fichero de configuración de Apache, `httpd.conf` y descomentar la directiva `LoadModule` correspondiente a `mod_ssl`:

```
LoadModule ssl_module modules/mod_ssl.so
```

3. Descomentar también las directivas de carga del fichero `httpd-ssl.conf` en el fichero `httpd.conf`:

```
# Secure (SSL / TLS) connections
Include conf/extra/httpd-ssl.conf
```

`Httpd-ssl.conf` incluye ya la directiva:

```
Listen 443
```

Por lo que en el fichero httpd.conf deja exclusivamente la del puerto 80.

```
Listen 80
```

4. Configurar las directivas de virtual hosting de Apache (fichero `httpd-ssl.conf`) para que el servidor por defecto escuche por el puerto 80 (manteniendo el `document_root`) y el servidor SSL escuche por el puerto 443 (se puede usar el mismo `document_root` o usar otro distinto si se quiere proteger sólo una parte de la web). Además, modificar las siguientes entradas del fichero `httpd-ssl.conf`:

```
SSLSessionCache none
#SSLSessionCacheTimeout 300
<VirtualHost *:443>
DocumentRoot "c:/Apache24/ssl"
<Directory "c:/Apache24/ssl">
  Options Indexes FollowSymLinks
  AllowOverride None
  Require all granted
</Directory>
ServerName localhost:443
ServerAdmin admin@example.com
ErrorLog "c:/Apache24/logs/ssl-error.log"
TransferLog "c:/Apache24/logs/ssl-access.log"
SSLEngine on
SSLCertificateFile "c:/Apache24/conf/server.crt"
SSLCertificateKeyFile "c:/Apache24/conf/server.pem"
</VirtualHost>
```

Hemos cambiado el DocumentRoot para que se vea bien el ejemplo.

5. Crear una página index.html en el directorio `document_root` del virtual host SSL para diferenciar los accesos:

```
<html>
  <body>
    <h1>Servidor SSL</h1>
    <p>Prueba de Servidor</p>
  </body>
</html>
```

6. Reiniciar el servidor Apache

7. Al conectarse a la URL `https://servidorprincipal`, en nuestro caso (https://localhost) se descargará el certificado (pedirá confirmación).

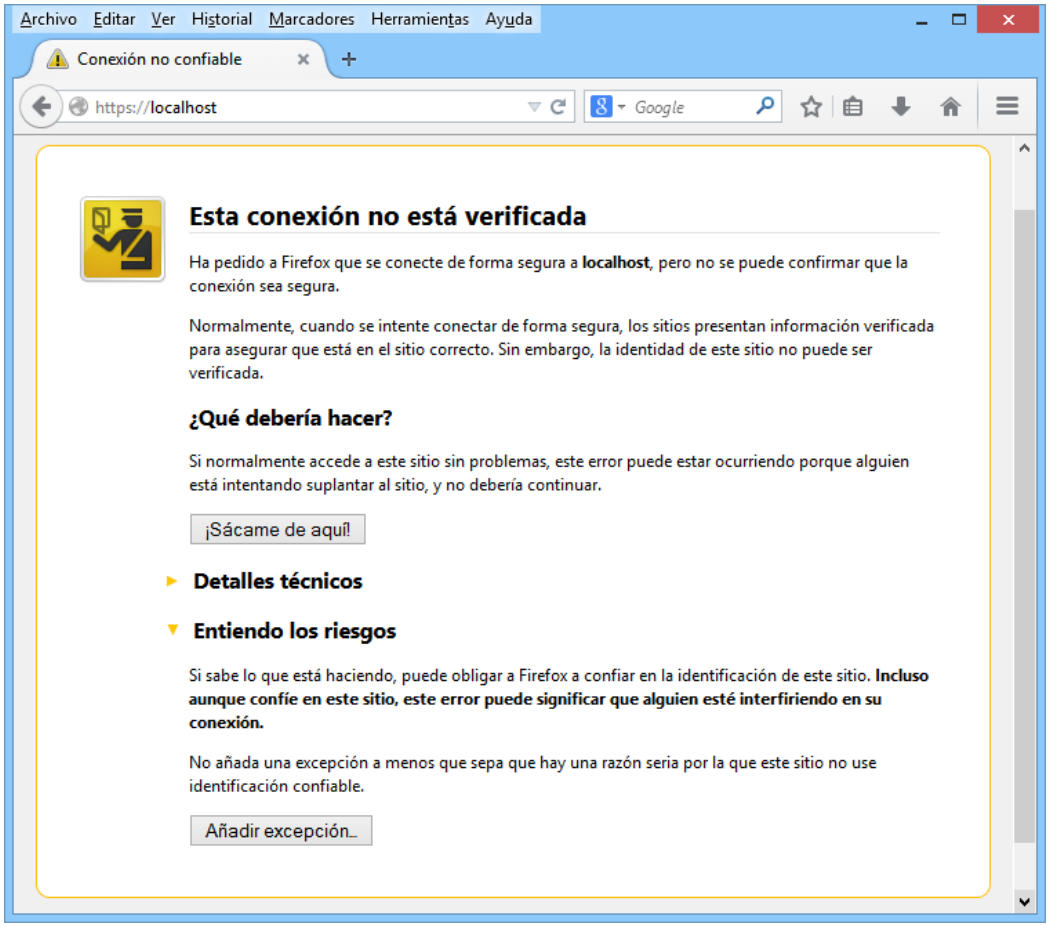

Figura 2.7.7: Certificado accediendo al servidor por https

Una vez confirmada, presentará la página `index.html` del `document_root` del virtual host SSL.

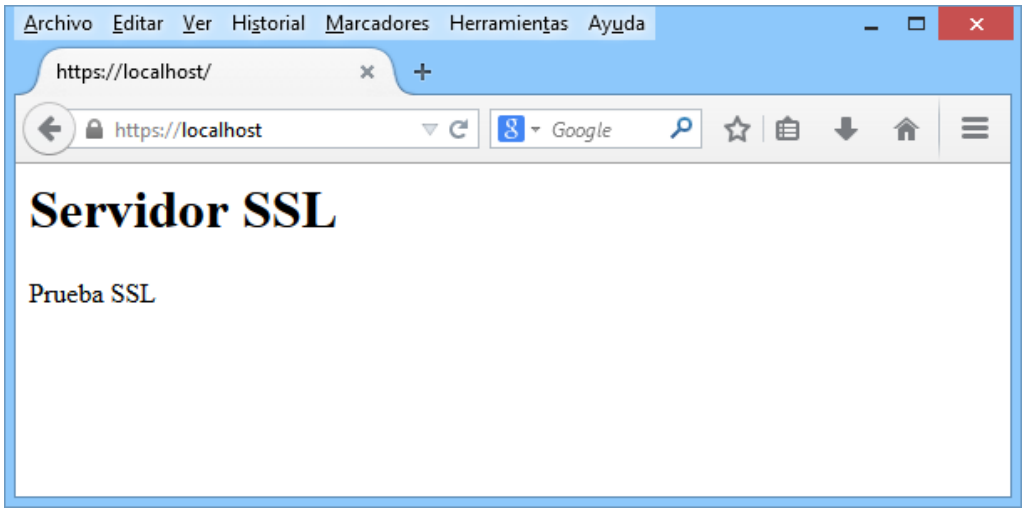

Figura 2.7.8: Acceso al servidor por https

8. Al conectarse a la URL `http://servidorprincipal`, (http://localhost) Apache devolverá la página inicial por defecto.

7.6. Conclusiones

- Apache puede incrementar grandemente sus posibilidades si se hace uso de la gran cantidad de módulos disponibles.
- Algunos de estos módulos forman parte de la distribución estándar y permiten, entre otras cosas, aumentar el rendimiento de Apache o hacer que Apache actúe como un sistema distinto (por ejemplo, como un Proxy).
- Se puede securizar el acceso a Apache mediante las directivas de autentificación y activando el soporte SSL, lo que permite que las comunicaciones entre el servidor y el cliente vayan cifradas.