Servidor Web

Preguntas

1. ¿En qué consiste HTTP? ¿y HTTPS? ¿Qué puertos y protocolo en la capa de transporte utilizan?

La función principal de un servidor web es dar páginas web a los clientes que las solicitan. Las páginas web son documentos escritos en un lenguaje de marcas llamado HTML. Dichas páginas html pueden tener asociadas imágenes, animaciones flash, etc. Cuando el navegador del cliente recibe el fichero con el texto de la página web, se encarga de solicitar al servidor web los ficheros adicionales con las imágenes y otro contenido asociado a dicha página. Puedes verificar dicho comportamiento des de el navegador Firefox → Herramientas del desarrollador → red , solicitando una página web.

El servidor indicará al cliente qué tipo de documento le está enviando en ese momento (página html, imagen jpeg, etc.) mediante un identificador estándar llamado "tipo mime".

Algunos servidores web, además de servir ficheros que los clientes les solicitan tal cual se encontraban en su disco duro (contenido estático), también tienen la habilidad de servir ficheros que no existían y que crean "al vuelo" cuando un cliente les solicita que ejecuten un programa (contenido dinámico).

Los clientes, además de solicitar páginas web estáticas o dinámicas al servidor web, también pueden enviar información a dicho servidor mediante formularios web.

El protocolo que el cliente y servidor web utilizan para comunicarse se llama HTTP, si la información no viaja cifrada, y HTTPS si la información viaja encriptada y el servidor web certifica ser quien dice ser (- interesante leer https://jvns.ca/blog/2017/01/31/whats-tls/ -).

Los servicios HTTP y HTTPS funcionan sobre TCP, utilizando por defecto los puertos 80 y 443, respectivamente.

¿Cómo funciona el servicio de páginas web?

(1) Imaginemos que el servidor web www.calico.com tiene una página web llamada /episodios/cap6.html con el siguiente contenido html:

```
<html>
  <body>
    <p>disponible en 2 días</p>
  </body>
</html>
```

(2) El cliente escribe la dirección de la página en el navegador para solicitarla al servidor web

```
http://www.calico.com/episodios/cap6.html
```

(3) Dicha petición se envía por internet como una solicitud HTTP

```
GET /episodios/cap6.html HTTP/1.1
Host: www.calico.com
User-Agent: Mozilla/5.0 (X11; Linux i686; ca) Ubuntu/9.10 Firefox/3.5.8
```

(4) El servidor va a buscar dicho fichero en la carpeta donde tenga alojada la web, por ejemplo:

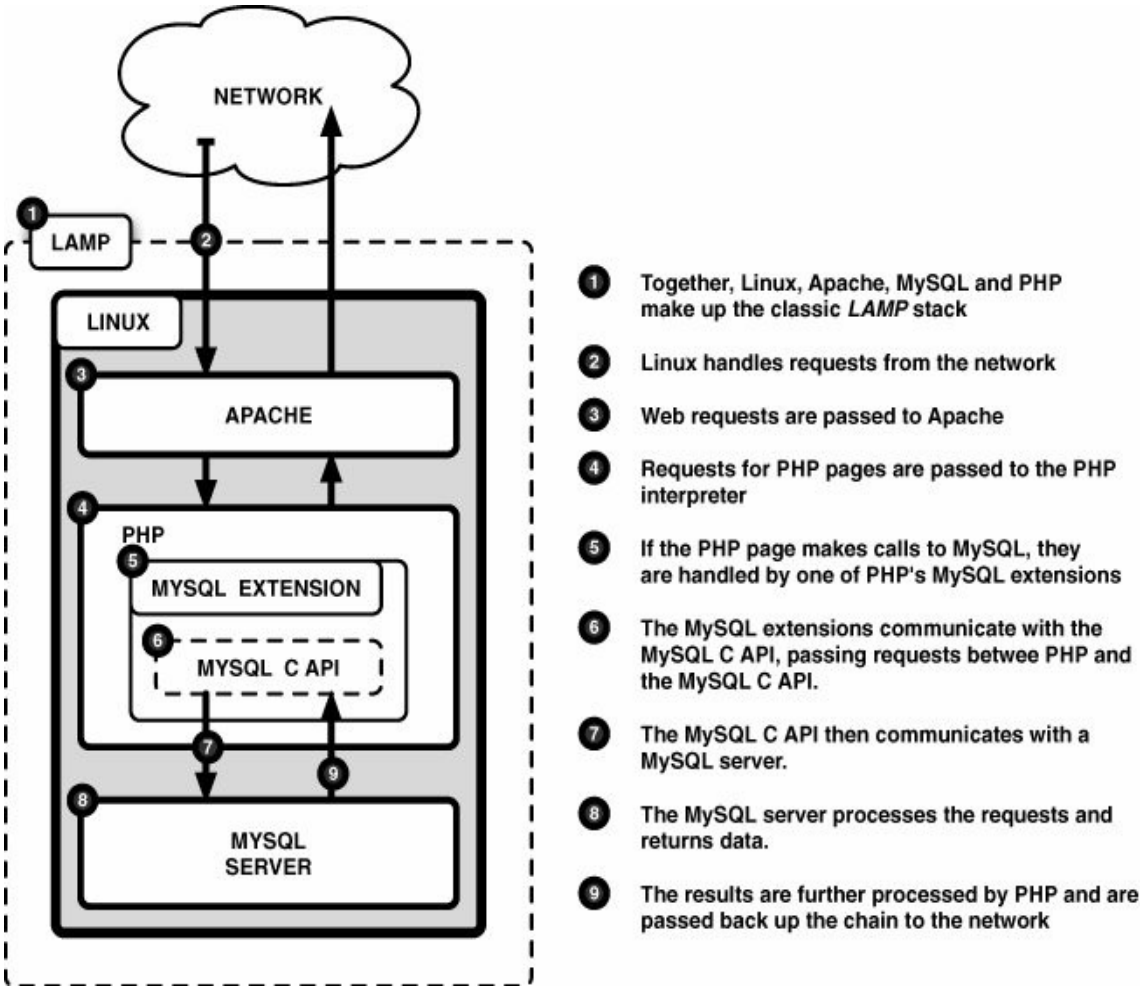
```
/var/www/webdecalico/episodios/cap6.html
```

(5) Dicho fichero se envía por internet como una respuesta HTTP

```
HTTP/1.0 200 OK
Date: Fri, 12 Mar 2020 15:45:04 GMT
Server: Apache/2.2.15 (Red Hat)
Content-Length: 65
Content-Type: text/html; charset=iso-8859-1
<html>
  <body>
    <p>disponible en 2 días</p>
  </body>
</html>
```

2. Dibuja el esquema de una red donde hay un servidor de HTTP sirviendo páginas a los ordenadores de la intranet. Dicho servidor HTTP se comunica con un servidor SQL de bases de datos para resolver páginas dinámicas ASP o PHP.

El servidor web dinámico podría estar en cualquier lugar de la red. Podría estar todo (servidor web + servidor de aplicaciones + servidor de bases de datos) repartido en varias máquinas, o en una única.



3. ¿Cuál es la sintaxis de la instrucción GET del protocolo HTTP? ¿Qué otras instrucciones existen en dicho protocolo?

La sintaxis es GET /carpetas/fichero HTTP/1.1

Las instrucciones de HTTP son HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, y CONNECT; pero solo nos interesaremos por GET para pedir páginas web, y POST para

enviar la información de formularios.

Podemos probar la instrucción GET mediante telnet a un sítio web por el puerto 80. Eso nos permitirá escribir comandos HTTP y estudiar la respuesta del servidor. Por ejemplo, prueba:

```
$ telnet www.xtec.cat 80
  GET / HTTP/1.1
  host:www.xtec.cat
```

4. En la petición de una página web mediante el protocolo HTTP, ¿qué tipos de respuestas son las indicadas por los códigos 1xx, 2xx, 3xx, 4xx y 5xx?

http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Códigos 1xx: respuesta provisional informativa → petición recibida, continuando proceso.

Códigos 2xx: todo correcto → petición recibida correctamente, entendida y aceptada.

Códigos 3xx: redirección → cliente debe tomar acción adicional para completar la petición.

Códigos 4xx: error del cliente → solicitud con sintaxis incorrecta o no puede procesarse.

Códigos 5xx: error del servidor → falló al completar una solicitud aparentemente válida.

5. En el protocolo HTTP ¿Cuáles son los cinco campos que te parecen más importantes en los "headers" de la petición de una página web y de la respuesta?

http://en.wikipedia.org/wiki/List_of_HTTP_headers

Hay muchos, todos interesantes. Algunas cabeceras que viajan con la petición son:

- Host: nombre del servidor al que se solicita la página.

- Cookie: envía el contenido de una cookie acompañando la petición.

- User-Agent: datos del navegador que realiza la petición (nombre, versión, idioma, sist.op.)

- Accept-...: tipo de información que acepta como respuesta (codificación, tipo documento, …)

- Content-Type y Content-Lenght: tipos de datos del formulario que se envían.

Algunas cabeceras que viajan con la respuesta son:

- Set-Cookie: envía una cookie con la respuesta.

- Server: nombre y versión del servidor web.

- Content-Encoding y Content-Lenght: tipo y longitud de los datos de respuesta que se envían.

6. Viene algún software de servidor de HTTP con Windows Server? ¿Desde dónde se instala y desde dónde se administra? ¿Y para Windows XP?

Con Windows Server viene el Internet Information Server (IIS).

Se instala desde "Agregar y quitar programas → Agregar y quitar componentes de Windows → Servidor de aplicaciones → Instalar Internet Information Services → Servicio World Wide Web → Servicio World Wide Web".

Se administra desde "Herramientas Administrativas → Internet Information Services"

7. ¿Cuál es el software de servidor de HTTP más conocido para Unix/Linux? ¿Y el software de

servidor más ligero de HTTP?

El servidor web más conocido és el flexible y potente *Apache*, però existen muchos otros servidores, como los rápidos y ligeros *NginX*, *Lighttpd*, y *Cherokee*.

8. Al instalar un servidor de HTTP, ¿Qué parámetros a configurar piensas que serán los más importantes?

- Puerto.

- Carpeta con las páginas web para cada host virtual.

- Límite de conexiones simultáneas.

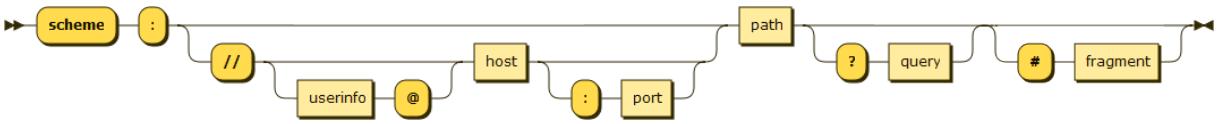
9. De qué maneras un servidor web puede servir páginas de más de un nombre de dominio ("virtual hosting"). Es decir, si necesitamos que un servidor web aloje las páginas de www.pepe.com y www.jose.com, ¿Cómo podemos conseguir qué distinga cuál le están pidiendo?

1) Asociamos cada nombre de dominio a un puerto diferente, donde escucha un servidor web. Por ejemplo: http://www.pepe.com:80/ y http://www.jose.com:81/ . Será incomodo para navegar.

2) Asociamos a la tarjeta de red del servidor web varias IPs (IPs virtuales). A continuación asociamos cada nombre de dominio a una de las IPs, donde escucha un servidor web. Por ejemplo: http://www.pepe.com/ será 192.168.15.2 y http://www.jose.com/ será 192.168.15.3 . Necesitaremos decirle al servidor DNS que resuelva www.pepe.com como 192.168.15.2 y www.jose.com como 192.168.15.3 .

3) Desde la versión 1.1 del protocolo HTTP, el cliente envía al servidor web junto con la petición de página web una cabecera indicando el nombre del servidor al que pide la página. Con ello el servidor web puede distinguir el host, y buscar la página dentro de la carpeta que contiene las páginas de dicho host.

10. ¿Cuál es la sintaxis completa de una URL?



Está especificado en el RFC 1738. Por ejemplo:

https://ana:pass@www.example.com:666/forum/questions/e.php?tag=networking&order=newest#top

Datos de la práctica

Nuestro servidor web servirá páginas de dos hosts: www.mired.org y intranet.mired.org, cuyos nombres deberemos tener integrados en el DNS (como alias del servidor *Asterix*).

Una vez funcionen ambos, en www.mired.org instalaremos el gestor de contenidos *MediaWiki*, que utilizaremos para ir guardando la documentación de las prácticas.

Generaremos un certificado digital y lo integraremos en el servidor web para que lo utilice en las

páginas seguras en intranet.mired.org .

Práctica con Windows

Instalaremos Internet Information Server para Windows Server. Exploraremos la interfaz gráfica de administración del servidor web, configurando los parámetros básicos para una pequeña red local, y probaremos el servidor.

Configuraremos el servidor web para que sirva páginas de dos nombres de dominio diferentes mediante tres métodos: puerto, ip, y nombre de host.

Paso a paso en Windows Server 2016 ó 2019:

- https://luigiasir.wordpress.com/2017/11/26/servidor-web-iis-en-windows-server-2016/
- https://www.jmsolanes.net/servidor-web-iis-windows-2016-asp-net/
- https://computingforgeeks.com/install-and-configure-iis-web-server-on-windows-server/
- https://computingforgeeks.com/how-to-configure-default-site-in-iis-server/

Práctica con Linux: página básica

Instalaremos Apache 2 para Linux. Exploraremos los ficheros de configuración. Crearemos una sencilla página web y navegaremos por ella desde un cliente.

1. Instalamos el servidor web *Apache* (alternativamente podríamos haber escogido otros servidores web como *NGINX* o *lighttpd*):

   ```
   # apt install apache2
   ```

   Comprobamos el estado del servidor:

   ```
   # systemctl status apache2
   # ss -plnt | grep apache
   ```

2. Exploramos los ficheros de configuración:

   ```
   # ls -l /etc/apache2
   ```

   - `/etc/apache2/apache2.conf` → fichero de configuración principal
   - `/etc/apache2/ports.conf` → fichero de configuración de puertos de escucha
   - `/etc/apache2/conf-available/` → fragmentos adicionales de configuración
   - `/etc/apache2/conf-enabled/` → fragmentos de configuración activados con `a2enconf`

- `/etc/apache2/mods-available/` → módulos (extensiones) disponibles para *Apache*
- `/etc/apache2/mods-enabled/` → módulos activados con `a2enmod` que se cargarán
- `/etc/apache2/sites-available/` → configuración de cada sitio web disponible
- `/etc/apache2/sites-enabled/` → configuración de cada sitio web activo, habilitada con `a2ensite`

Las carpetas `(sites/mods/conf)-enabled` tan solo contienen enlaces simbólicos a ficheros que están en las carpetas `(sites/mods/conf)-available` respectivas.

Otros sitios y ficheros interesantes:

- `/var/www/html/` → alojamiento de las páginas web por defecto
- `/var/log/apache2/access.log` → registro de peticiones de páginas web
- `/var/log/apache2/error.log` → registro de errores del servidor web

Recomiendo leer http://www.control-escape.com/web/configuring-apache2-debian.html

3. Pero antes que nada crearemos el sitio web con una página de prueba:

   # **`cat etc/apache2/sites-enabled/000-default.conf`**

   nos muestra la configuración del único sitio habilitado, e indica que las páginas de dicho sitio se encuentran en /var/www/html. Así pués, cuando naveguemos por http://servidor/ estaremos pidiendo ficheros de dicha carpeta. Vamos a crear una subcarpeta con una nueva página:

   # **`mkdir /var/www/html/miweb`**

   # **`nano /var/www/html/miweb/index.html`**

   ```
   <html><body><p>Mi primera página</p></body></html>
   ```

   Comprobemos que funciona navegando por http://*servidor*/miweb/

   Es importante tener en cuenta los permisos lectura en las carpetas y ficheros que creamos para la web: **¡Si el servidor web no tiene permiso de lectura sobre ellos, no los podrá servir.** El servidor web trabaja como usuario *www-data*. Para acceder a las páginas web debería tener permisos de lectura:

   # **`chmod -R a+r /var/www/`**

   O bien ser propietario de las páginas web:

   # **`chown -R www-data:www-data /var/www/`**

Antes de continuar debemos saber como gestionar los problemas. Si hemos realizado cambios en la configuración de *Apache* y no consigue reiniciar, podemos consultar mensajes de error de tres maneras diferentes:

`journalctl -xe`

`cat /var/log/apache2/error.log`

`apache2ctl configtest`

Práctica con Linux: virtual hosts

"Virtual Hosts" es el mecanismo que tiene un servidor web que le permite servir varios sitios con diferente nombre de dominio. Por ejemplo, un mismo servidor web sirviendo páginas de www.mired.org y de www.iam.cat. Cada sitio o dominio tendrá asociada una carpeta con su correspondientes páginas web, y el servidor web distinguirá entre sitios utilizando la cabecera "host" de las peticiones HTTP 1.1. Puedes aprender más sobre dominios virtuales en *Apache* en http://httpd.apache.org/docs/current/vhosts/name-based.html

Un posible esquema de ejemplo de virtual hosts en *Apache* seria:

```
<VirtualHost *:80>
  ServerName www.dominio_1.algo
  ServerAlias otros_nombres_host
  DocumentRoot /carpeta1_html/
</VirtualHost>

<VirtualHost *:80>
  ServerName www.dominio_2.algo
  DocumentRoot /carpeta2_html/
</VirtualHost>
```

Puedes encontrar más ejemplos en http://httpd.apache.org/docs/current/vhosts/examples.html, y los errores más comunes en https://cwiki.apache.org/confluence/display/HTTPD/CommonMisconfigurations.

1. Creamos una web y un host virtual para wiki.mired.org:

   # **mkdir /var/www/html/wiki**

   # **nano /var/www/html/wiki/index.html**

   ```
   <html><body><p>Mi wiki</p></body></html>
   ```

   # **nano /etc/apache2/sites-available/wiki.conf**

   ```
   <VirtualHost *:80>
     ServerName wiki.mired.org
     DocumentRoot /var/www/html/wiki
   </VirtualHost>
   ```

   # **a2ensite wiki**

   # **systemctl reload apache2**

2. Creamos una web y un host virtual para intranet.mired.org:

   # **mkdir /var/www/html/intranet**

   # **nano /var/www/html/intranet/index.html**

   ```
   <html><body><p>Intranet</p></body></html>
   ```

   # **nano /etc/apache2/sites-available/intranet.conf**

   ```
   <VirtualHost *:80>
     ServerName intranet.mired.org
   ```

```
  DocumentRoot /var/www/html/intranet
</VirtualHost>
```

a2ensite intranet

systemctl reload apache2

3. Por último, para que los clientes de la red reconozcan el nombres de los hosts, intranet.mired.org y wiki.mired.org, editamos el archivo de configuración de nuestro servidor de DNS:

   # **nano /etc/bind/db.mired.org**

```
asterix         A          192.168.100.2
wiki            CNAME      asterix
intranet        CNAME      asterix
```

   # **systemctl reload bind9**

   Si no tenemos servidor de DNS, podemos hacer un “apaño” en el cliente con su fichero *hosts* (https://en.wikipedia.org/wiki/Hosts_(file)). Dicho fichero se utiliza para resolver nombres de màquina en Internet antes que la petición DNS. En Linux dicho fichero está en /etc/hosts y en Windows está en c:\windows\system32\drivers\etc\hosts . En el cliente modificaremos:

   # **nano /etc/hosts**

```
192.168.100.2        wiki.mired.org
192.168.100.2        intranet.mired.org
```

   # **ping wiki.mired.org**

4. Ya solo falta comprobar que funciona desde un cliente navegando por http://intranet.mired.org/ y http://wiki.mired.org/

Práctica con Linux: páginas seguras con HTTPS

Para la creación de la página web segura deberemos seguir tres pasos: (1) Habilitar el módulo de HTTPS para *Apache*; (2) Generar un certificado; y (3) Configurar nuestra página para que utilice dicho certificado.

Ver: https://wiki.debian.org/Self-Signed_Certificate

1. Para habilitar el módulo de HTTPS de *Apache* debemos escribir:

   # **a2enmod ssl**

   # **systemctl restart apache2**

   Comprobemos que el servidor web escucha por el puerto 443:

   # **ss -plnt | grep apache**

2. Para que nuestro servidor intranet.mired.org pueda servir páginas seguras con el protocolo HTTPS, necesita un certificado. Dicho certificado permitirá que nuestro servidor utilice cifrado asimétrico para intercambiar las claves de cifrado simétrico con los clientes, antes de iniciar una trasmisión segura de información.

   El formato del certificado está especificado por el estándar X.509 y normalmente son emitidos por una entidad llamada "Autoridad Certificadora" (CA). Inicialmente un cliente o navegador sólo acepta los certificados de CA en las que confía.

   Sin embargo, el cliente deberá aceptar el certificado directamente del servidor, ya que generaremos un certificado "autofirmado". Si queremos evitarlo, deberíamos comprar un certificado a una entidad certificadora confiable, pero tiene un precio que no merece la pena pagar en un entorno educativo.

   Si quieres aprender un poco más sobre certificados en *Apache* puedes visitar la página http://httpd.apache.org/docs/current/ssl/ , especialmente las partes "FAQ/aboutcerts" y la documentación de referencia del módulo SSL.

   Para generar nuestro certificado autofirmado en el directorio */etc/ssl/certs*, ejecutaremos el comando `openssl` (muchas opciones) o el comando `make-ssl-cert` (que es una "envoltura" más simple del anterior).

   Generarlo rápidamente con `make-ssl-cert` sería:

   ```
   # make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/certs/intranet.pem
   ```

   Si en cambio hubieras querido generar el certificado paso a paso con `openssl` sería:

   ```
   # openssl genrsa 2048 > /etc/ssl/private/intranet.key
   # chown root:ssl-cert /etc/ssl/private/intranet.key
   # chmod 640 /etc/ssl/private/intranet.key
   # openssl req -new -x509 -nodes -sha1 -days 365 -key
       /etc/ssl/private/intranet.key > /etc/ssl/certs/intranet.pem
   ```

   Si quieres ver la información del certificado, ejecuta:

   ```
   # cat /etc/ssl/certs/intranet.pem
   # openssl x509 -in /etc/ssl/certs/intranet.pem -inform PEM -text
   ```

   (Des de hace unos años se puede conseguir un certificado gratuito de *Let's Encrypt*, que es confiable. El proceso para generarlo y utilizarlo lo puedes leer en la siguiente página: https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-debian-10)

3. Ahora nos falta editar el fichero de configuración de Apache para informar de que el sitio intranet.mired.org utiliza cifrado mediante el certificado generado. Para ello crearemos un host adicional:

   ```
   # nano /etc/apache2/sites-available/intranet.conf
   ```

   ```
   # Host virtual https://intranet.mired.org/
   <VirtualHost *:443>
     ServerName intranet.mired.org
     DocumentRoot /var/www/intranet
     SSLEngine On
     SSLCertificateFile /etc/ssl/certs/intranet.pem
   # Depende del formato del certificado necesitaremos o no lo siguiente
   # SSLCertificateKeyFile /etc/ssl/private/intranet.key
   ```

```
</VirtualHost>

# Si queremos redirigir http://intranet.mired.org/ a https
<VirtualHost *:80>
  ServerName intranet.mired.org
  Redirect permanent / https://intranet.mired.org
</VirtualHost>
```

```
# a2ensite intranet
# systemctl reload apache2
```

Práctica con Linux: páginas web personales para usuarios del sistema

Si queremos activar las carpetas web de los usuarios de nuestro sistema, deberemos cargar el módulo *userdir* de *Apache:*

```
# a2enmod userdir
# systemctl reload apache2
```

Por defecto, la carpeta web de cada usuario estará en */home/usuario/public_html* y se accederá navegando por *http://servidor_web/~usuario/*

Fuera bueno que la carpeta web *public_html* de los usuarios y sus ficheros asociados tengan los permisos 755. Si queremos que dicha carpeta se genere automáticamente, la podemos crear en el directorio */etc/skel* .

Práctica con Linux: HTTP/2

Si queremos activar que algún sitio web trabaje con el protocolo HTTP/2 en lugar de HTTP/1.1 deberemos seguir los siguientes pasos:

1. Habilitar los siguientes módulos de *Apache*:

```
# a2enmod ssl
# a2enmod http2
# systemctl restart apache2
```

2. Modificar el fichero de configuación principal de *Apache*:

```
# nano /etc/apache2/apache2.conf
```

```
Protocols h2 http/1.1
```

```
# systemctl reload apache2
```

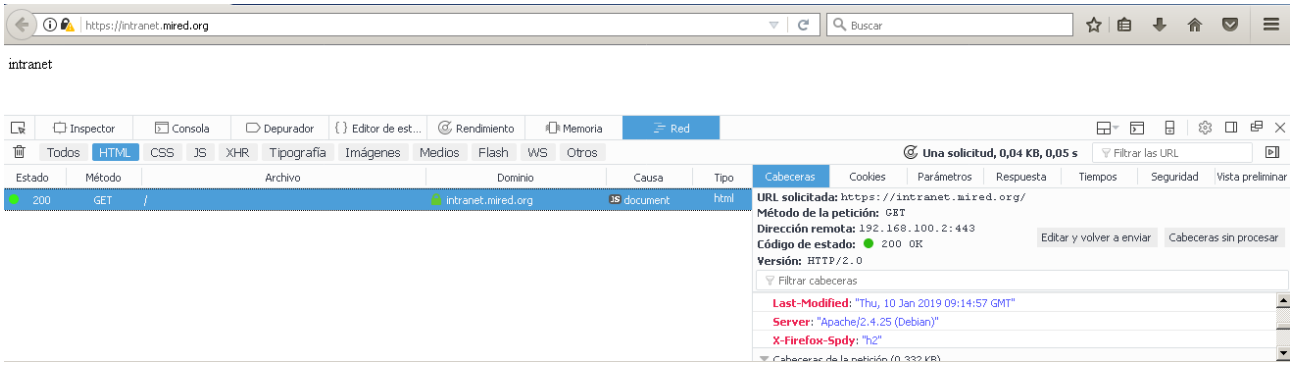
3. Modificar el fichero de configuación del sitio web que utilizará HTTP/2. En nuestro caso podemos utilizar uno de un ejercicio anterior, por ejemplo “intranet”:

```
# nano /etc/apache2/sites-available/intranet.conf
```

```
<VirtualHost *:443>
  ...
  Protocols h2 http/1.1
</VirtualHost>
```

```
# systemctl reload apache2
```

4. Verifica el comportamiento des de el navegador Firefox → Herramientas del desarrollador → red , solicitando una página web → click en la petición → cabeceras:



Práctica con Linux: servidor dinámico con PHP y MariaDB

Instalaremos el lenguaje de programación PHP, y el servidor de bases de datos relacionales MariaDB para Linux. Integraremos Apache y PHP, para que el servidor pueda ejecutar páginas dinámicas php.

1. Suponemos que *Apache* ya está instalado. En caso contrario ejecuta:

```
# apt install apache2
```

2. Instalamos el intérprete de PHP, y el módulo de Apache para que sepa gestionar páginas php:

```
# apt install php libapache2-mod-php
```

```
# systemctl restart apache2
```

```
# ls -l /etc/apache2/mods-enabled/ | grep php
```

Comprueba que PHP queda integrado creando un pequeño fichero de prueba:

```
# nano /var/www/html/index.php
```

```
<?php phpinfo(); ?>
```

Y ahora navega sobre http://servidor_web/index.php

Una manera mucho más detallada de comprobar que se han integrado *Apache* y PHP sería indagando en el contenido de algunos ficheros:

En */etc/apache2/mods-enabled/php7.4.load* se carga el módulo de PHP para *Apache*:

```
LoadModule php7_module /usr/lib/apache2/modules/libphp7.4.so
```

En */etc/apache2/mods-enabled/php7.4.conf* se asocia la extensión .php al módulo de PHP:

```
<FilesMatch ".+\.php$">SetHandler application/x-httpd-php</Filesmatch>
```

En */etc/apache2/mods-enabled/dir.conf* se dice cuales son las páginas por defecto:

```
DirectoryIndex index.html index.php ...
```

Y en */etc/mime.types* se asocia la extensión .php a un tipo de fichero:

```
application/x-httpd-php phtml pht php
```

Una opción interesante, para mejorar el rendimiento de las páginas dinámicas, sería instalar adicionalmente una caché de PHP como *Opcache*, o una caché genérica como *Memcached*:

```
# apt install php7.4-opcache php-apcu
```

3. Instalamos el servidor de BBDD relacionales *MariaDB* (fork de *MySQL*):

   ```
   # apt install mariadb-server php-mysql
   ```

   Configuraremos la instalación. En la segunda pregunta responde con la nueva contraseña. Al resto de preguntas escoge la respuesta por defecto (-pero mucho mejor si las lees e intentas entenderlas-):

   ```
   # mysql_secure_installation
   ```

   Comprobamos el estado de MariaDB / MySQL, y que escucha peticiones por el puerto 3306:

   ```
   # systemctl status mysql
   # ss -plnt
   ```

   Alternativamente a la instalación de *MariaDB*, pudiéramos haber escogido otro servidor de bases de datos como *PostgreSQL*:

   ```
   # apt install postgresql php-pgsql
   ```

4. Instalaremos *phpMyAdmin* para crear y administrar bases de datos remotamente desde un navegador web:

   Veremos cómo instalar *phpMyAdmin* des de la fuente en Internet, aunque en Debian 11 la instalación des de repositorios es mucho más sencilla, escribiendo tan sólo:

   ```
   # apt install phpmyadmin
   ```

Alternativamente a la instalación de *phpMyAdmin*, pudiéramos haber escogido el gestor web de bases de datos *Adminer*, que se encuentra en los repositorios, o el gestor web de bases de datos *DBeaver*, que se puede descargar des de Internet:

```
# apt install adminer
```

En primer lugar lo descargamos, descomprimimos, lo movemos a /usr/share , y nos aseguramos que el usuario que ejecuta Apache tenga propiedad o permisos sobre los ficheros.

```
# wget https://files.phpmyadmin.net/phpMyAdmin/5.1.1/phpMyAdmin-5.1.1-all-languages.tar.gz
# tar xvf phpMyAdmin-5.1.1-all-languages.tar.gz
# mv phpMyAdmin-5.1.1-all-languages/ /usr/share/phpmyadmin
# chown -R www-data:www-data /usr/share/phpmyadmin
```

En segundo lugar creamos la base de datos para phpMyAdmin:

```
# cat /usr/share/phpmyadmin/sql/create_tables.sql | mariadb -u root -p
```

En tercer lugar, instalamos alguna dependencia:

```
# apt install php-xml
```

En cuarto lugar editaremos el fichero de configuración de phpMyAdmin:

```
# cp /usr/share/phpmyadmin/config.sample.inc.php /usr/share/phpmyadmin/config.inc.php
# nano /usr/share/phpmyadmin/config.inc.php
```

```
$cfg['blowfish_secret'] = 'mi_secreto';
```

En quinto lugar indicaremos a Apache como acceder al sitio phpMyAdmin:

```
# nano /etc/apache2/sites-available/phpmyadmin.conf
```

```
Alias /phpmyadmin /usr/share/phpmyadmin
```

```
# a2ensite phpmyadmin
# systemctl reload apache2
```

Prueba si funciona navegando desde un cliente por http://servidor/phpmyadmin .

Te deberás identificar como un usuario del servidor de bases de datos, y los que puedas hacer dependerá de los permisos del usuario de la base de datos con el que inicias sesión.

Práctica con Linux: instalando Mediawiki des de repositorios

1. Instalaremos el gestor de contenidos *MediaWiki* (http://www.mediawiki.org) :

   # **apt install mediawiki**

   Dicha instalación generará las páginas web dinámicas en el directorio */var/lib/mediawiki*, que en realidad serán enlaces a ficheros en */usr/share/mediawiki*. También generará un fichero */etc/apache2/conf-enabled/mediawiki.conf* que es una copia de */etc/mediawiki/mediawiki.conf*. Si listas el contenido de dicho fichero, verás que *Apache* lo reconoce como web a servir gracias al comando *Alias* de */etc/mediawiki/apache.conf*:

   # **nano /etc/mediawiki/apache.conf**

   ```
   Alias /mediawiki /var/lib/mediawiki
   ```

2. Deberíamos poder navegar por http://*servidor*/mediawiki/

   La primera vez que navegues, se lanzará el proceso de instalación de *MediaWiki*, que creará una base de datos, un usuario propietario de la base de datos, un usuario administrador del wiki, etc. Para ello seguramente nos pedirá la contraseña del usuario administrador del servidor SQL. Por último nos pedirá que movamos el fichero de configuración en alguna parte:

   # **cp LocalSettings.php /etc/mediawiki/**

3. Para poder subir ficheros e imágenes al wiki necesitaremos comprobar esta línea en el fichero de configuración:

   # **nano /etc/mediawiki/LocalSettings.php**

   ```
   $wgEnableUploads = true;
   ```

4. Para comenzar a editar páginas en el *MediaWiki*, nos será de utilidad alguna chuleta con las etiquetas de formato: http://upload.wikimedia.org/wikipedia/commons/a/ac/Cheatsheet-es.pdf

Práctica con Linux: instalando gestores de contenido

En caso de querer instalar algunos gestores de contenido de forma manual, que es la más recomendada, aquí dejo una serie de pasos para *Mediawiki*, *Joomla*, *Drupal* y *Wordpress*. No he probado las siguientes instrucciones a fondo, así que pueden contener algún error.

Recuerda que cuando se instala un gestor de contenidos, la primera vez que se navega sobre éste, se realiza la configuración. En dicha configuración el gestor de contenido necesita crear las tablas que contendrán su información en el servidor de bases de datos. Para ello, o bien os preguntará por el nombre y contraseña del usuario administrador del servidor de bases de datos, o bien os habrá obligado antes a crear una base de datos y un nuevo usuario local con todos los permisos sobre dicha base de datos.

1. **Joomla**:

   ```
   # cd ~
   # wget https://downloads.joomla.org/es/cms/joomla4/4-0-5/Joomla_4-0-5-Stable-Full_Package.tar.gz
   # mkdir /var/www/html/joomla
   # tar -xzvf Joomla_4-0-5-Stable-Full_Package.tar.gz -C /var/www/html/joomla
   # chown -R www-data:www-data /var/www/html/joomla
   # apt install php-curl php-gd php-intl php-xml php-zip
   # mysql -u root -p
      MariaDB > CREATE DATABASE joomla_db;
      MariaDB > CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'passwd';
      MariaDB > GRANT ALL ON joomla_db.* TO 'usuario'@'localhost';
      MariaDB > EXIT;
   ```

   Ahora navega por http://*servidor*/joomla y comenzará el proceso de instalación. Una vez configurado, navega por http://*servidor*/joomla/administrator para administrar *Joomla*.

2. **Mediawiki**:

   ```
   # cd ~
   # wget https://releases.wikimedia.org/mediawiki/1.37/mediawiki-1.37.1.tar.gz
   # mkdir /var/www/html/mediawiki
   # tar -xzvf mediawiki-1.37.1.tar.gz -C /var/www/html/mediawiki
   # chown -R www-data:www-data /var/www/html/mediawiki
   ```

```
# chmod a+w /var/www/html/mediawiki/config
# apt install php-curl php-gd php-intl php-xml php-zip
# mysql -u root -p
   MariaDB > CREATE DATABASE mediawiki_db;
   MariaDB > CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'passwd';
   MariaDB > GRANT ALL ON mediawiki_db.* TO 'usuario'@'localhost';
   MariaDB > EXIT;
```

Y ahora navega por http://*servidor*/mediawiki y comenzará el proceso de instalación. Una vez configurado, movemos el fichero de configuración.

```
# mv /var/www/html/mediawiki/config/LocalSettings.php
/var/www/html/mediawiki/
# chmod o-rw /var/www/html/mediawiki/LocalSettings.php
# chmod a-w /var/www/html/mediawiki/config
```

3. **Wordpress**:

```
# cd ~
# wget https://wordpress.org/latest.tar.gz
# mkdir /var/www/html/wordpress
# tar -xzvf latest.tar.gz -C /var/www/html/wordpress
# chown -R www-data:www-data /var/www/html/wordpress
# apt install php-curl php-gd php-intl php-xml php-zip
# mysql -u root -p
   MariaDB > CREATE DATABASE wordpress_db;
   MariaDB > CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'passwd';
   MariaDB > GRANT ALL ON wordpress_db.* TO 'usuario'@'localhost';
   MariaDB > EXIT;
```

Y ahora navega por http://*servidor*/wordpress y comenzará el proceso de instalación. Una vez configurado, navega por http://*servidor*/wordpress/wp-login para administrar *Wordpress*.

4. **Drupal**:

```
# cd ~
# wget https://www.drupal.org/download-latest/tar.gz
# mkdir /var/www/html/drupal
# apt install php-curl php-gd php-intl php-xml php-zip
# tar -xzvf drupal-9.3.2.tar.gz -C /var/www/html/drupal
# chown -R www-data:www-data /var/www/html/drupal
# mysql -u root -p
   MariaDB > CREATE DATABASE drupal_db;
   MariaDB > CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'passwd';
   MariaDB > GRANT ALL ON drupal_db.* TO 'usuario'@'localhost';
```

```
MariaDB > EXIT;
```

Y ahora navega por http://*servidor*/drupal y comenzará el proceso de instalación. Una vez configurado, si quieres poner *Drupal* en castellano, desde un navegador descarga el paquete de *Drupal* en español que está en la página web https://localize.drupal.org/translate/languages/es , o bien puedes ejecutar en el servidor el comando:

```
# cd ~
# wget https://ftp.drupal.org/files/translations/all/drupal/drupal-9.3.2.es.po
```

Y sigue los siguientes pasos:

- En la web de *Drupal* que has instalado, ves al apartado de administración de módulos y activa el módulo *Locale*.

- En la web de *Drupal* que has instalado, ves al apartado "Configuración del sitio → Localización → importar idioma", y ahí seleccionas el archivo que deseas importar, que es el que te has descargado, seleccionando la casilla de idioma español.

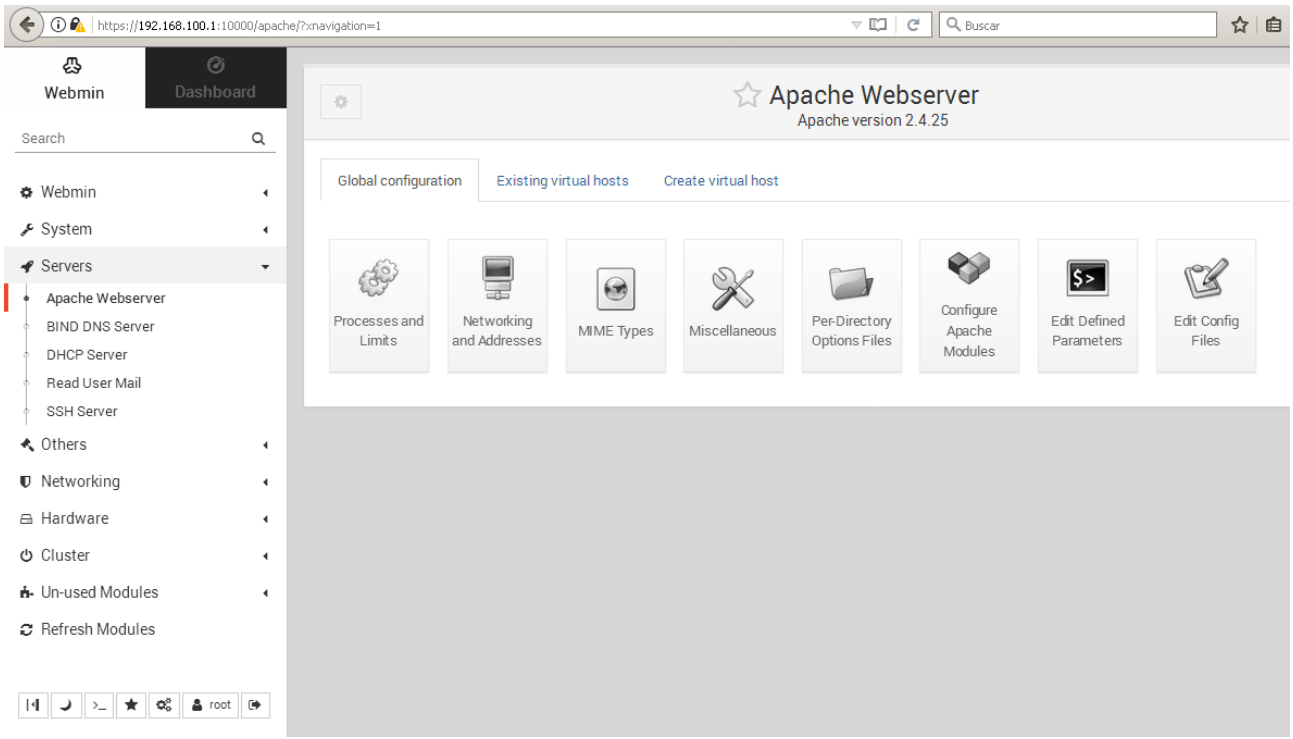
- En la web de *Drupal* que has instalado, ves al apartado "Configuración del sitio → Localización → añadir idioma", y añades el español. A continuación guarda la configuración.

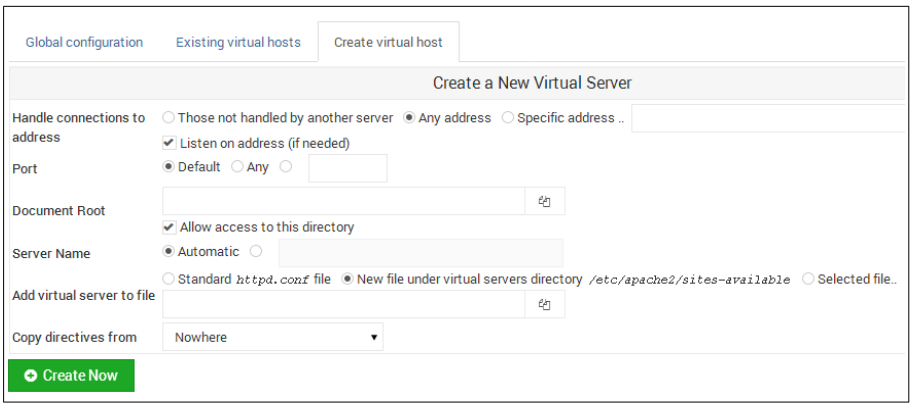
Práctica con Linux: administración con herramientas gráficas (webmin)

https://*IP_servidor*:10000/ → Un-used modules → Apache Webserver

https://*IP_servidor*:10000/ → Refresh modules

https://*IP_servidor*:10000/ → Servers → Apache Webserver





Práctica con Linux: seguridad en Apache

1. Actualitzar Apache de manera regular

2. Esconder la versión de Apache para que no aparezca en las cabecera “server” de la respuesta HTTP.

   # **nano /etc/apache2/apache2.conf**

   ```
   ServerSignature Off
   ServerTokens Prod
   ```

   # **systemctl restart apache2**

3. Inhabilitar módulos innecesarios con `a2dismod`:

   # **a2dismod mod_status**

   # **a2dismod mod_info**

   …

   # **systemctl restart apache2**

4. Analizar periódicamente logs de Apache en busca de peticiones que han fallado y peticiones extrañas. Por ejemplo:

   # **grep "client denied" error_log | tail -n 10**

5. Cuidado con enlaces simbólicos en las carpetas con páginas web, que no nos lleven fuera de la ruta deseada. Por ejemplo: `ln -s / public_html` . Revisa las siguientes opciones en los ficheros de configuración del sitio web:

   `Options -FollowSymLinks`

   `Options +SymlinksIfOwnerMatch`

6. No dejar listar un directorio que no tiene página index.html o index.php u otras por defecto.

Revisa las siguientes opciones en los ficheros de configuración del sitio web:

`Options -Indexes` dentro de `<Directory …>`

7. No ejecutes el servidor web como administrador, sino con el usuario restringido del sistema que suele venir predeterminado cuando se instala dicho servidor.

   Cuando un programa se ejecuta, las cosas que puede hacer sobre el sistema dependen de los privilegios del usuario que lo ejecutó. Esto es muy importante en un servidor web que puede ejecutar contenido dinámico, ya que el usuario a quien proporcionamos alojamiento web puede subir ficheros php con código para hacer cualquier cosa.

8. Ves con cuidado con todo el contenido ejecutable. Vigila los CGI y los SSI.

   ```
   Options IncludesNOEXEC
   ```

9. Ves con cuidado con los ficheros .htaccess . Dichos ficheros, que se guardan ocultos en las carpetas web, contienen configuración adicional para dicha web, que se aplica a las páginas de dicha carpeta y subcarpetas. Así pues, un usuario a quien proporcionamos alojamiento web puede subir ficheros .htaccess a sus carpetas para sustituir la configuración original que dimos al web.

   Podemos configurar qué directivas se pueden anular mediante .htaccess . En un caso extremo podemos especificar que ninguna se pueda cambiar:

   ```
   AllowOverride None
   ```

   También podemos configurar que los ficheros .htaccess no se descargen o lean:

   ```
   <Files ".ht*">
     Require all denied
   </Files>
   ```

10. Puedes forzar que el acceso a ciertas partes de la web sea autentificado con nombre de usuario y contraseña, poniendo un fichero ,htaccess en la carpeta correspondiente:

    ```
    Require valid-user
    AuthName "Private directory"
    AuthType Basic
    AuthUserFile /etc/apache2/authfiles/htpasswd-private
    ```

    En el ejemplo anterior la contraseña y el contenido de la carpeta viajan sin cifrar. El fichero *AuthUserFile* contiene los nombres de usuario y contraseñas, y se manipula con el comando `htpasswd`

11. Puedes restringir el acceso a la web bajo ciertas condiciones con la directiva `require`

12. Puedes instalar los módulos *mod_security* y *mod_evasive* para proteger el servidor web de ataques de fuerza bruta y de ataques de denegación de servicio, respectivamente.

13. Intenta evitar los ataques de denegación de servicio que se pueden producir con peticiones demasiado grandes, controlando cuanta información pueden subir las peticiones y formularios

con `LimitRequestBody`, `LimitRequestFields`,y `LimitRequestFieldSize`. Por ejemplo:

```
<Directory "/var/www/myweb1/user_uploads">
  LimitRequestBody 512000
</Directory>
```

14. Minimiza ataques de denegación de servicio, como *Slowloris*, con las opciones:
    - `TimeOut` , `KeepAliveTimeout` , `RequestReadTimeout`
    - `MaxClients` , `MaxRequestWorkers` , `MaxKeepAliveRequests`

15. Protegerse de ataques cross-site scripting, con las opciones:

```
Header set X-XSS-Protection "1; mode=block"
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

16. Otros: puede ser interesante poner reglas en el cortafuegos que no permitan conexiones salientes des del servidor web, por si toman el control de él limitar lo que pueden hacer en la red.

17. Otros: puede ser interesante protegerse del aumento consumo de RAM por “pérdidas” constantes de memoria de un proceso del servidor web mal programado, matando dichos procesos cuando han atendido muchos clientes y creando nuevos.

```
MaxConnectionsPerChild 10000
```

Referencias

- http://en.wikipedia.org/wiki/Web_servers
- http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://en.wikipedia.org/wiki/Media_type
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP
- http://news.netcraft.com/archives/web_server_survey.html
- http://en.wikipedia.org/wiki/Comparison_of_web_server_software
- https://jvns.ca/blog/2017/01/31/whats-tls/
- http://www.xtec.net/~acastan/textos/Tuning%20LAMP.pdf
- https://blog.avast.com/create-a-secure-web-server-avast
- https://mythemeshop.com/blog/web-server-security/