# Programación de Sistemas Distribuidos Práctica 3: Servidor Web, Apache

**Arturo Requejo Portilla**

UNIVERSIDAD
NEBRIJA

# Tabla de contenido

# I. Enunciados

A. Ejercicio 1
   Instalación de Apache 2.4
B. Ejercicio 2
   Por defecto se instalará en el directorio /var/www. ¿Qué puedes explicar de la estructura de directorios?
C. Ejercicio 3
   Crear 2 VirtualHosting nombre_apellido_1.nebrija.es y nombre_apellido_2.nebrija.es
D. Ejercicio 4
   ¿Qué configuración debes hacer en el servidor de nombres de tú portátil/ordenador?
E. Ejercicio 5
   ¿Eres capaz de añadir autenticación básica a tú servidor? ¿Puedes añadir usuario y contraseña distintos para cada virtual host?
F. Ejercicio 6
   Si realizas una petición GET a uno de los 2 virtual host. Detalla las cabeceras que recibes de la solicitud.
G. Ejercicio 7
   ¿Cómo ampliarías esta práctica?

# II. Introducción

Apache es un servidor web que se encarga de almacenar, procesar y servir las páginas web a los usuarios de estas. Se distribuye bajo una licencia de código abierto, lo que quiere decir que es gratuito y fácilmente adaptable, personalizable y reutilizable. No es un servidor físico, sino un software que se ejecuta en un servidor HTTP.

La función de Apache es la de servir de enlace entre el servidor y los navegadores de los usuarios del sitio web, mientras intercambia información entre ellos. Sigue una arquitectura cliente-servidor, que se caracteriza, por su centralización y escalabilidad.

El servidor Apache funciona de tal forma que cuando un cliente hace una solicitud al servidor, apache se encargara de responder con la respuesta correspondiente. Cuando atiende muchas solicitudes de forma simultánea permite responder a todas las solicitudes mediante archivos HTML, convirtiendo así todas las solicitudes que no tienen por qué estar en el mismo lenguaje a HTML.
Si se produce cualquier tipo de error Apache también se encargará de enviar esta respuesta al cliente.

De esta manera, el servidor y el cliente se comunican a través del protocolo HTTP y Apache actúa como el medio responsable de que esa comunicación sea fluida y segura, contribuyendo así a la optimización web.

Las principales ventajas de apache son:

- Al ser tan utilizado hay mucha documentación.
- Es multiplataforma siendo compatible con varios sistemas operativos
- Permite también soportar varios lenguajes de programación
- Es simple de instalar y configurar
- Cuenta con módulos de autenticación y acceso siendo muy seguro
- Gran flexibilidad

# III. Resultados

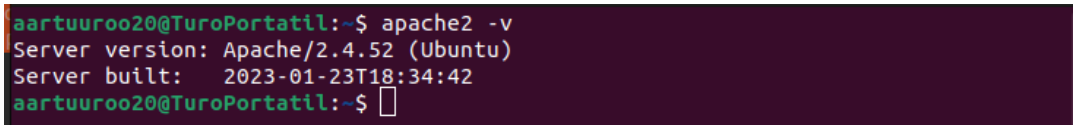
## A. Ejercicio 1

Para realizar la instalación del servidor apache 2.4 en el sistema operativo Linux, en la terminal realice el siguiente comando:

***sudo apt install apache2***

Una vez realizado este comando mire en la terminal si se había instalado correctamente y tenía la versión adecuada, para poder comprobar eso en la terminal de Linux ejecuté el comando:

***apache2 -v***




En la imagen superior podemos comprobar como mediante la ejecución del comando previamente descrito vemos como Apache está instalado en la versión 2.4.

## B. Ejercicio 2

Linux se basa en UNIX y, por tanto, toma prestada su jerarquía de sistemas de archivos de UNIX. Teniendo una estructura de directorios similar en los sistemas operativos tipo UNIX como BSD y macOS.

Linux sigue un Estándar de Jerarquía del Sistema de Archivos (FHS). Esto define la estructura de directorios y el contenido/propósito de los directorios en las distribuciones de Linux.

Todos los archivos y directorios en Linux se encuentran bajo la ‘raíz’ representada por ‘/’. Como todos los demás directorios o archivos descienden de la raíz, la ruta absoluta de cualquier archivo pasa por la raíz. Los directorios más famosos de Linux son:

- /bin -> Contiene directamente los archivos ejecutables de muchos comandos básicos del Shell.
- /dev -> Contiene archivos especiales, incluidos los relativos a los dispositivos. Son archivos virtuales, no están físicamente en el disco.
- /etc -> Contiene los archivos de configuración principales del sistema, utilizados principalmente por el administrador y los servicios, como el archivo de contraseñas y los archivos de red.
- /usr -> Todos los archivos ejecutables, las bibliotecas, el código fuente de la mayoría de los programas del sistema. Por esta razón, la mayoría de los archivos que contiene es de sólo lectura.
- /home -> Contiene los directorios personales de los usuarios. El directorio personal contiene los datos del usuario y los archivos de configuración específicos del usuario.
- /lib -> Códigos que pueden ser utilizados por los binarios ejecutables.
- /sbin -> Es similar al directorio /bin. La única diferencia es que contiene los binarios que sólo pueden ser ejecutados por root o un usuario sudo.
- /tmp -> Contiene archivos temporales.
- /var -> Contiene archivos de datos variables y temporales como por ejemplo los registros del sistema (logs), los registros de programas que tenemos instalados en el sistema operativo, archivos spool, etc. La principal función del directorio /var es la detectar problemas y solucionarlos.

## C. Ejercicio 3

Para el ejercicio 3 he seguido varias guías y he tenido que resolver una serie de problemas que explicaré más adelante. Primero empezare diciendo un comando que me ha servido de gran utilidad que es el cómo eliminar directorios que tengan un contenido dentro, el comando es el siguiente:

***sudo rm -r nombre_directorio***

Para empezar este ejercicio he tenido que crear los 2 dominios donde se almacenara toda información del html de nuestro servidor. Para ello he accedido primero desde la terminal de Linux al directorio /var/www, mediante el comando:

***cd /var/www***

Una vez estemos en el directorio debemos crear los subdirectorios que contendrán los previamente descrito, para ello mediante el siguiente comando:

***sudo mkdir -p /var/www/arturo_requejo_1_nebrija.es/public_html***

```
aartuuroo20@TuroPortatil:/var/www$ sudo mkdir -p /var/www/arturo_requejo_1.nebrija.es/public_html
aartuuroo20@TuroPortatil:/var/www$ ls
arturo_requejo_1.nebrija.es  html
aartuuroo20@TuroPortatil:/var/www$ sudo mkdir -p /var/www/arturo_requejo_2.nebrija.es/public_html
aartuuroo20@TuroPortatil:/var/www$ ls
arturo_requejo_1.nebrija.es  arturo_requejo_2.nebrija.es  html
aartuuroo20@TuroPortatil:/var/www$
```

Como vemos en la imagen superior hemos creado ambos directorios con sus nombres correspondientes donde gracias al comando ***ls*** podemos visualizar que archivos hay dentro del directorio www de la carpeta var.

Una vez hayamos creado los directorios debemos dar permisos a los mismos para que así estos puedan acceder y modificar los archivos al ser un usuario *no root,* este comando se debe realizar dos veces al tener dos directorios. También deberemos asignar permisos de lectura al directorio general para que así las páginas puedan ser visualizadas correctamente. Los comandos son los siguientes:

- Asignación de permisos a los directorios:

  ***sudo chown -R $USER:$USER /var/www/ arturo_requejo_1_nebrija.es /public_html***

- Asignación de permisos de lectura al directorio general:

  ***sudo chmod -R 755 /var/www***

```
aartuuroo20@TuroPortatil:/var/www$ ls
arturo_requejo_1.nebrija.es  arturo_requejo_2.nebrija.es  html
aartuuroo20@TuroPortatil:/var/www$ sudo chown -R $USER:$USER /var/www/arturo_requejo_1.nebrija.es/public_html
aartuuroo20@TuroPortatil:/var/www$ sudo chown -R $USER:$USER /var/www/arturo_requejo_2.nebrija.es/public_html
aartuuroo20@TuroPortatil:/var/www$ sudo chmod -R 755 /var/www
aartuuroo20@TuroPortatil:/var/www$
```

Una vez hayamos creado ambos directorios y asignado permisos deberemos crear los dos virtuales hostings en nuestros correspondientes directorios. Empezaremos abriendo la terminal y escribiendo un comando que nos va a crear en el directorio escrito un archivo html para nuestra página web, utilizaré como editor de texto “nano”, aunque podremos utilizar entornos más visuales como visual studio.

***nano /var/www/arturo_requejo_1.nebrija.es/public_html/index.html***

Este comando lo tenemos que realizar también en el otro directorio, en mi caso es arturo_requejo_2.nebrija.es. Para poder crear el segundo archivo html podemos copiar la configuración del index.html del directorio 1, para ello usaremos el comando:

***cp /var/www/arturo_requejo_1.nebrija.es /public_html/index.html /var/www/arturo_requejo_2.nebrija.es /public_html/index.html***

Posteriormente abriremos este segundo archivo creado previamente y realizaremos las modificaciones que creamos convenientes.

```
aartuuroo20@TuroPortatil:/var/www$ nano /var/www/arturo_requejo_1.nebrija.es/public_html/index.html
aartuuroo20@TuroPortatil:/var/www$ cp /var/www/arturo_requejo_1.nebrija.es/public_html/index.html /var/www/arturo_requejo_2.nebrija.es/public_html/index.html
aartuuroo20@TuroPortatil:/var/www$ nano /var/www/arturo_requejo_2.nebrija.es/public_html/index.html
```

El paso final de este ejercicio es realizar la creación de nuestros archivos virtual host. Estos archivos van a dictar la configuración de nuestros hosts y como el servidor Apache responderá a varias peticiones de dominio. Empezaremos copiando el archivo que apache trae por defecto para la configuración de host, este archivo lo modificaremos posteriormente con una serie de atributos.

***sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/arturo_requejo_1.nebrija.es.conf***

Con el comando introducido en la parte superior crearemos un archivo de virtual host en el dominio arturo_requejo_1.nebrija.es con las mismas características que el archivo por defecto de apache.

Una vez creado el archivo entraremos dentro del mismo donde deberemos especificar:

- Server admin: Deberemos especificar un correo a donde llegara información al administrador del estado del servidor.
- Nombre del servidor: Nombre de cómo se reconocerá nuestro servidor
- Alias del servidor: Como se puede reconocer a nuestro servidor si no ponemos el nombre exacto.
- Document root: Deberemos indicar donde se encuentra el directorio

***sudo nano /etc/apache2/sites-available/ arturo_requejo_1.nebrija.es.conf***

```
 GNU nano 6.2        /etc/apache2/sites-available/arturo_requejo_1.nebrija.es.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin arequejop@alumnos.nebrija.es
        ServerName arturorequejo1.com
        ServerAlias www.arturorequejo1.com
        DocumentRoot /var/www/arturo_requejo_1.nebrija.es/public_html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn
```

```
aartuuroo20@TuroPortatil:/var/www$ sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/arturo_requejo_1.nebrija.es.conf
aartuuroo20@TuroPortatil:/var/www$ sudo nano /etc/apache2/sites-available/arturo_requejo_1.nebrija.es.conf
aartuuroo20@TuroPortatil:/var/www$ sudo cp /etc/apache2/sites-available/arturo_requejo_1.nebrija.es.conf /etc/apache2/sites-available/arturo_requejo_2.nebrija.es.conf
aartuuroo20@TuroPortatil:/var/www$ sudo nano /etc/apache2/sites-available/arturo_requejo_2.nebrija.es.conf
aartuuroo20@TuroPortatil:/var/www$ ls
arturo_requejo_1.nebrija.es  arturo_requejo_2.nebrija.es
aartuuroo20@TuroPortatil:/var/www$
```

Una vez hayamos realizado la primera configuración deberemos realizar los mismos pasos para el segundo directorio creado, creando así su respectivo archivo de configuración. Cambiando los campos de nombre de servidor, servidor alias y document root poniendo la respectiva información al segundo archivo.

## D. Ejercicio 4

Para este ejercicio 4 debemos explicar que cambiamos debemos realizar en el servidor de nombres de nuestro portátil para que el servidor Apache pueda funcionar de manera local. Primero debemos habilitar los archivos de nuestro virtual host, para poder habilitarlo hay varias herramientas que nos permiten esto, pero yo las habilitaré con a2ensite.

Antes de empezar voy a hacer una breve explicación de que es a2ensite, a2ensite es una script que habilita un sitio especificado dentro de la configuración de Apache 2, lo hace creando enlaces simbólicos dentro de una ruta apache, también permite deshabilitar estos enlaces simbólicos.

El comando que se mostrara a continuación permite la habilitación de los archivos de host los directorios especificados.

***sudo a2ensite arturo_requejo_1.nebrija.es.conf***

Este comando lo escribiremos dos veces con nuestros archivos correspondientes, gracias a este comando habilitaremos ambos archivos.

```
aartuuroo20@TuroPortatil:/var/www$ sudo a2ensite arturo_requejo_1.nebrija.es.conf
Enabling site arturo_requejo_1.nebrija.es.
To activate the new configuration, you need to run:
  systemctl reload apache2
aartuuroo20@TuroPortatil:/var/www$ sudo a2ensite arturo_requejo_2.nebrija.es.conf
Enabling site arturo_requejo_2.nebrija.es.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Otro comando interesante es: ***sudo a2dissite arturo_requejo_1.nebrija.es.conf***
Este comando nos facilita el poder deshabilitar la página que previamente habíamos abierto.

```
aartuuroo20@TuroPortatil:/var/www$ sudo a2dissite arturo_requejo_1.nebrija.es.conf
[sudo] password for aartuuroo20:
Site arturo_requejo_1.nebrija.es disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
aartuuroo20@TuroPortatil:/var/www$ sudo a2dissite arturo_requejo_2.nebrija.es.conf
Site arturo_requejo_2.nebrija.es disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Un paso importante es comprobar si la sintaxis del código html es correcta o si hay algún error en la configuración del archivo, para ello utilizaremos el comando:

***sudo apache2ctl configtest***

Este comando solo es necesario si no estamos utilizando ningún entorno desarrollo integrado como puede ser visual studio ya que nos facilitan ver fallo de sintaxis.

Por último, como bien pone en la terminal debemos reiniciar apache para que los cambiamos que hayamos realizado se guarden, para poder iniciar o reiniciar el servidor utilizaremos el comando:

***sudo systemctl restart apache2***

```
aartuuroo20@TuroPortatil:/var/www$ sudo systemctl restart apache2
aartuuroo20@TuroPortatil:/var/www$
```

Una vez hayamos habilitado los archivos de del host virtual deberemos realizar una serie de cambios en el servidor de nombre local de nuestro ordenador. Si estamos en un sistema operativo Linux deberemos acceder a la terminal y escribir el siguiente comando:

***sudo nano /etc/hosts***

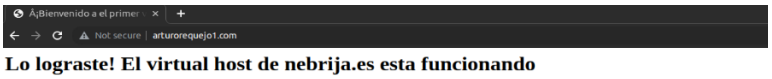
```
 GNU nano 6.2                                  /etc/hosts
127.0.0.1       localhost
127.0.1.1       TuroPortatil

127.0.0.1  arturorequejo1.com
127.0.0.1  arturorequejo2.com

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Este comando nos permitirá acceder a los archivos de host de nuestro ordenador Linux donde deberemos modificar el archivo y escribir, primero la dirección IP de nuestro ordenador y seguido de ello el nombre del servidor de dominio que obviamente debe coincidir con el nombre de servidor que hemos configurado el en cada archivo host.

Ya para finalizar podemos comprobar que nuestro servidor Apache funciona yendo a cualquier navegador y escribir en la url:

***http://arturorequejo1.com* o *http://arturorequejo2.com***

El nombre de dominio cambiara dependiendo del nombre del servidor que hayamos configurado.




**Lo lograste! El virtual host de nebrija.es esta funcionando**

## E. Ejercicio 5

Para realizar la autenticación en Apache se puede realizar de dos formas. Realizarla por el propio html o por el propio servidor de Apache.

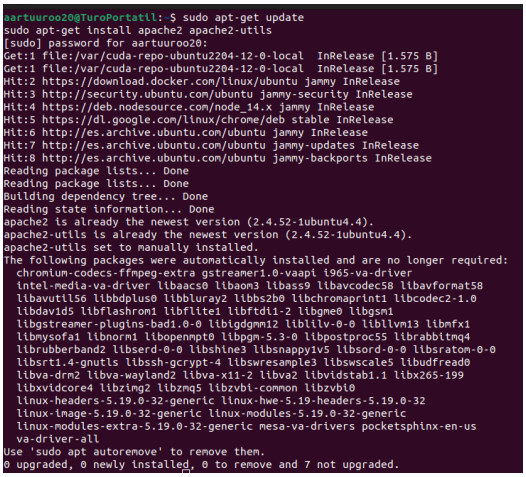
Para realizarla por html seria introducir dos inputs:

- Un input donde introdujeses el nombre de usuario.
- El otro para introducir la contraseña.
- También haría falta un botón para enviar estos valores y posteriormente ya la propia página permite al usuario entrar o no.

Para realizarla por el servidor Apache la explicare a continuación, esta forma es la que yo he realizado por lo tanto explicare como hacerla detalladamente.

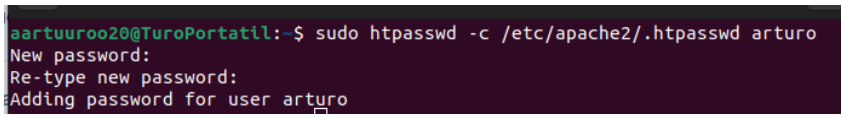
Vamos a empezar creando un archivo el cual va a almacenar a los usuarios con sus respectivas contraseñas encriptadas. Para ello empezaremos instalando la siguiente librería de Apache que nos permitirá realizar esto.

***sudo apt-get install apache2 apache2-utils***




Una vez instalada empezaremos creando el archivo mencionado anteriormente, donde con el siguiente comando, especificaremos tanto nombre de usuario y contraseña.

***sudo htpasswd -c /etc/apache2/.htpasswd username***

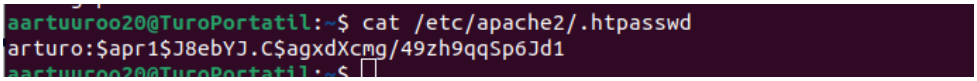



La -c solo se pondrá cuando tengamos que crear este archivo por primera vez, si quisiésemos añadir más usuarios bastaría con escribir el siguiente comando.

***sudo htpasswd /etc/apache2/.htpasswd another_user***

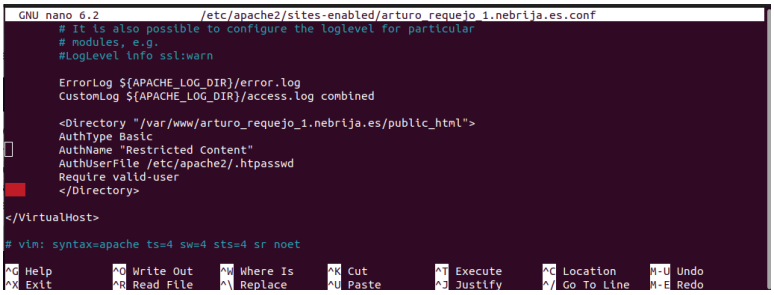
Una vez hayamos creado los usuarios que queramos podemos comprobar si estos se han creado correctamente mediante este comando.

***cat /etc/apache2/.htpasswd***




Como vemos en la captura superior mediante el comando cat abrimos un archivo en la ruta especificada y en este caso nos muestra el usuario con la contraseña asociada encriptada.

Una vez realizado este deberemos configurar nuestros servidores de Apache para implementar la autenticación. Para ello deberemos volver a acceder al archivo de configuración del servidor e incluir las siguientes líneas de código.




Para acceder al fichero de configuración deberemos realizar el comando:

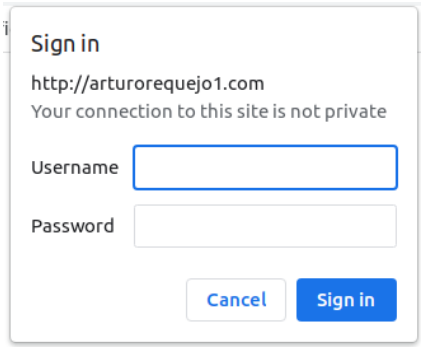
***sudo nano /etc/apache2/sites-enabled/arturo_requejo_1.nebrija.es.conf***

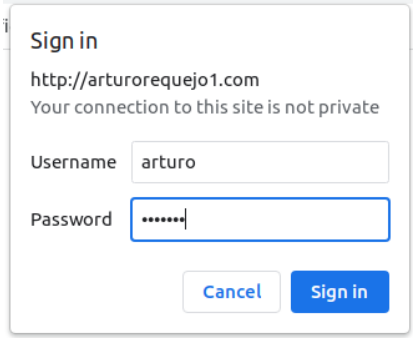
Las líneas de código que añadimos significan lo siguiente:

- En el directorio deberemos especificar donde queremos implementar la autenticación. En mi caso será a la hora de acceder al html de mis servidores web.
- AuthType, especificaremos el tipo de autenticación que queramos, en nuestro caso pondremos una básica de nombre y usuario.
- AuthName, escribiremos el mensaje que queremos que nos salga a la hora de introducir las credenciales.
- Authuserfile, especificaremos donde tiene que ir el servidor Apache para comprobar las credenciales introducidas.
- Require valid-user, significa que cualquiera que pueda verificar su identidad con su contraseña puede entrar.

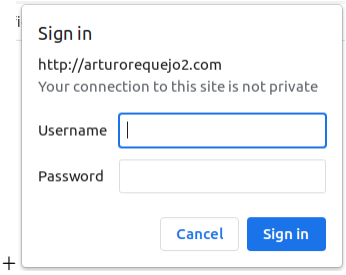
Por último, solo queda reiniciar el servidor de Apache para implementar los cambios. Mediante el comando:

***sudo service apache2 restart***



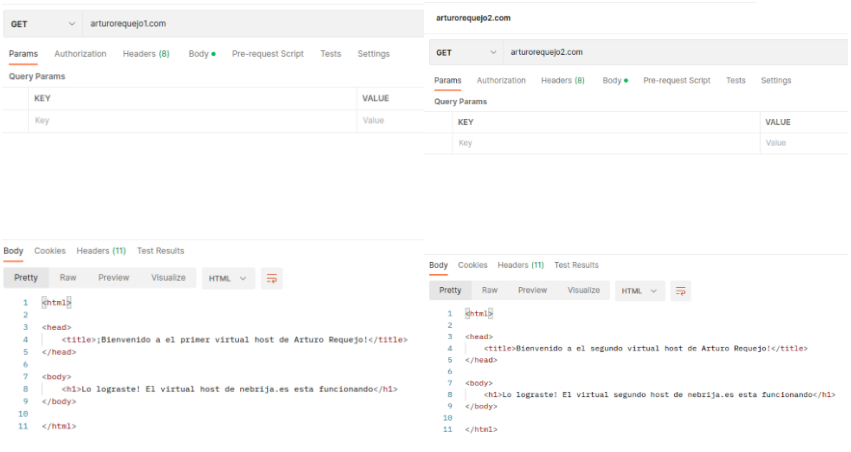

Como vemos en las imágenes superiores al entrar al dominio, arturorequejo1.com nos pide las credenciales de acceso, si ponemos correctamente el nombre de usuario y contraseña nos dejara acceder. Este paso lo deberemos repetir una vez más para configurar el otro dominio.



## F. Ejercicio 6

Para poder realizar este ejercicio he utilizado la herramienta Postman. Postman es una aplicación que sirve para el desarrollo de APIs, esta aplicación con interfaz de usuario nos permite realizar peticiones http, siendo estas GET, POST, UPDATE y DELETE, a una API y gracias a estas solicitudes podemos ver las respuestas de nuestra página.

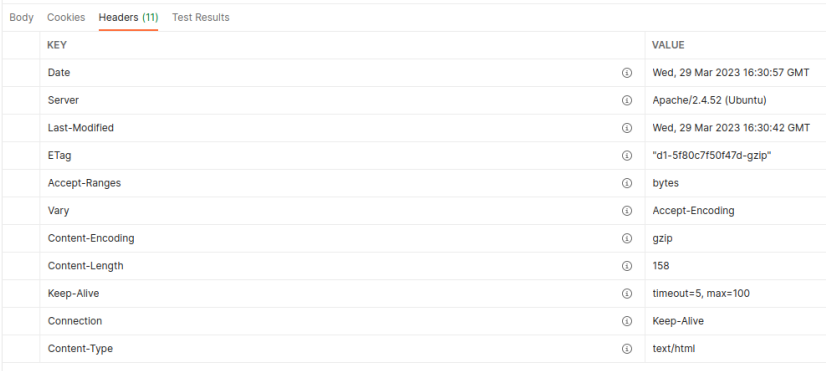
Una petición GET nos permite recuperar recursos de un servidor web obteniendo así datos de nuestro servidor especificado a treves de la url. Los parámetros de consulta se envían a traces de la url pudiendo buscar así parámetros específicos. No modifica los datos del servidor, sino que solo solicita que se devuelvan esos datos solicitados.




Como vemos en las imágenes superiores hemos realizado una petición GET a cada servidor web realizado. Primero empezamos eligiendo el tipo de petición que queremos, posteriormente ponemos la url al servidor web el cual queremos hacer la petición y en el body recibiéremos la respuesta de este.

A continuación, explicaré los que son los headers de un servidor web. Los headers son parte de las respuestas de HTTP que se envían desde el servidor web al cliente en respuesta a una solicitud proporcionando así información adicional sobre la respuesta y el servidor web que la envía. También pueden utilizarse para establecer ciertas características de seguridad o control.

Al acceder a los headers de la solicitud previamente realizada obtenemos lo siguiente:



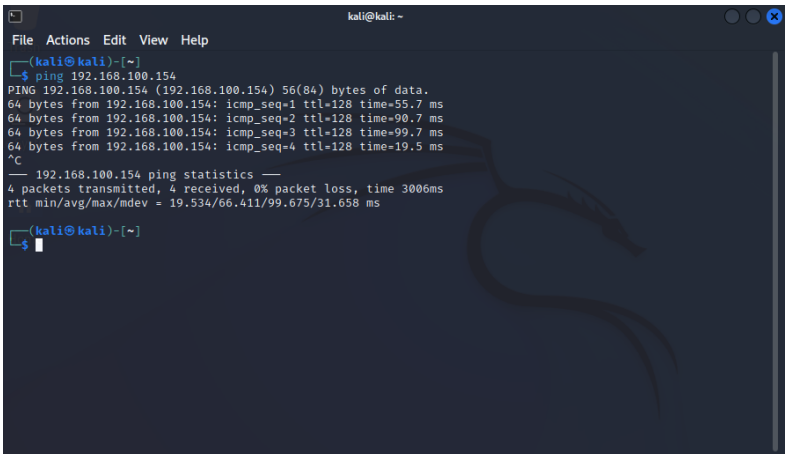
Estos están formados por una clave y un valor, yendo de arriba abajo obtendremos lo siguiente:

- Date -> Indica la fecha la cual se realizó la petición.
- Server -> Indica el tipo de servidor al cual se accedió.
- Last-Modified -> Indica la última actualización que se realizó en el servidor.
- Etag (entity tag) -> Identificador de una versión especifica a un recurso, permite a las caches ser más eficiente y salvar ancho de banda.
- Accept-Ranges -> Indica la compatibilidad con solicitudes del cliente para la descarga de archivos e indica también la unidad para el rango.
- Vary -> Se utiliza para llevar una trazabilidad de los headers que han sido importantes en la generación de la respuesta.
- Content-Encoding -> Permite la codificación del mensaje o payload.
- Content-Lenght -> Indica el tamaño total de la entidad de la respuesta en decimal.
- Keep-Alive -> El remitente establece un tiempo de espera y una cantidad máxima de solicitudes.
- Connection -> Comprueba si la conexión permanece abierta cuando la transacción haya acabado, en este caso es persistente y no cerrada permitiendo realizar solicitudes posteriores al mismo servidor.
- Content-Type -> Indica el tipo medio original del recurso.

## G. Ejercicio 7

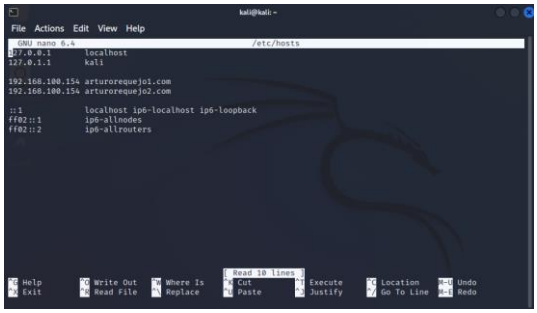
Una forma de poder ampliar esta práctica es comprobando si la red Apache es segura tal como la hemos realizado y si las credenciales del usuario se envían correctamente. Para ello utilice una aplicación de Linux llamada Wireshark, el cual es una aplicación que analiza protocolos e intercambio de mensajes que pasan a través de una red wifi.

Al tener Linux Ubuntu y no Kali Linux probé incluso a ampliar un poco esta idea, quería conectar mi ordenador con el servidor Apache cual cualquier dispositivo conectado a mi red local. Me descargué una máquina virtual Kali Linux en mi ordenador sobremesa y conecté todos mis ordenadores a la misma red. Una vez conectado realice una prueba de conexión desde mi máquina virtual al ordenador portátil.

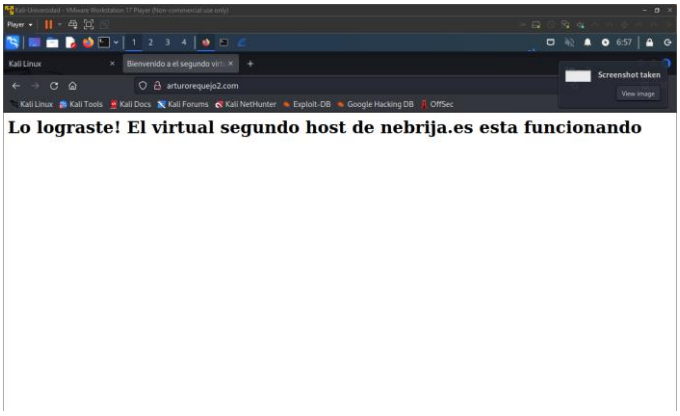


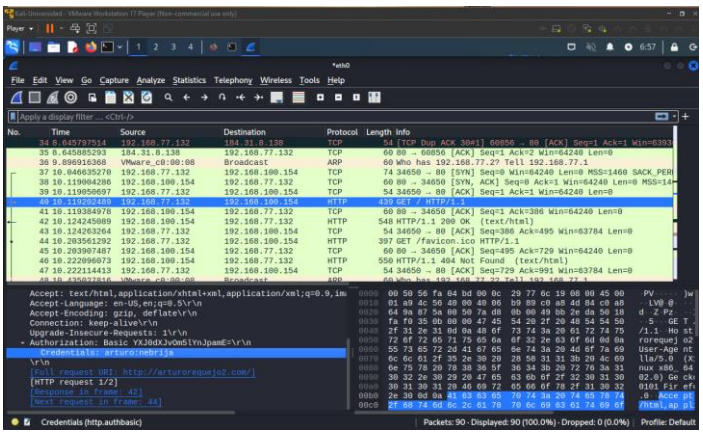

Una vez habiendo visto que tenemos conexión a el ordenador tuve que acceder al archivo de configuración de host de la máquina virtual y añadir tanto la dirección IP de mi ordenador portátil y el propio nombre del servidor al cual acceder.

***sudo nano /etc/hosts***




Para finalizar solo basta con abrir tanto un navegador web y Wireshark y realizar una prueba de tanto acceso a la página como análisis de red.




En la imagen superior podemos comprobar cómo se está realizando la petición de acceso desde la máquina virtual al servidor Apache localizado en el otro ordenador. También podemos comprobar como el servidor pide las credenciales al cliente y el cliente las manda. Vemos como claramente aparece tanto el usuario como la contraseña. Esto supone que cualquiera con un sniffer pueda analizar la red y coger los datos de autenticación de cualquiera que intenta acceder al servidor.

La forma en la que se podría solucionar esto, es realizando una encriptación de las credenciales para así evitar robos de información o suplantación de identidad.

Otro ejercicio de ampliación para esta practica seria permitir a otros ordenadores localizados en otras redes conectarse a la pagina web. Para ello deberíamos acceder a la configuración de nuestro router y abrir el puerto 80, que es donde el servidor está a la escucha. Aunque es más recomendable definir un puerto externo e interno para así tener una mayor seguridad. Por último, en la máquina a la cual queremos acceder, bastaría con poner la ip publica seguido del puerto definido previamente.

## IV. Conclusiones

Esta tercera practica de Apache me ha servido para adquirir conocimientos de cómo funciona un servidor Apache y como configurarlo. También como realizar configuraciones de autenticación y como permitir a otros dispositivos de la misma red local poder acceder al servidor y para finalizar he podido profundizar en la estructura de directorios y ficheros de Linux.

## V. Bibliografía


- https://phoenixnap.com/kb/how-to-edit-hosts-file-in-windows-mac-or-linux
- https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-20-04
- https://manpages.debian.org/jessie/apache2/a2ensite.8.en.html
- https://developer.mozilla.org/es/docs/Web/HTTP/Headers
- https://www.digitalocean.com/community/tutorials/how-to-set-up-password-authentication-with-apache-on-ubuntu-14-04