



TRABAJO RETO1 UD1

# **Ejecución de código tras un servidor web**

**Asignatura: DWES**

**Alumnos: Carolain Maccha, Luis Ayllon, Arancha  
Chicharro.**

## Reto 1 UD1 Ejecución de código tras un servidor web

**Objetivo:** entender la diferencia entre páginas estáticas y páginas generadas dinámicamente. en el funcionamiento de un servidor web.

**Introducción:** antes de meternos con PHP vamos a ver la ejecución de código en cualquier lenguaje detrás de un servidor web con la tecnología CGI, que, aunque no es recomendable en producción nos sirve como prueba de concepto para entender lo que vendrá después.

**Desarrollo:** en grupos de dos o tres alumnos, se trata de empezar a experimentar con un servidor web (Apache) mediante los siguientes pasos:

\*\*\*\*    \*\*\*\*    \*\*\*\*

*Aprovechando que cada compañero tenemos sistemas operativos diferentes, todo el ejercicio lo hemos hecho tanto en el Sistema Operativo de Kali Linux como en Ubuntu.*

### 1. Instalación del servidor web Apache de acuerdo a lo visto en despliegues.

**S.O. kali Linux:** Verificamos que en el S.O kali linux ya lo tenemos instalado.

```
(kali㉿kali)-[~]  
$ sudo apachectl -v  
[sudo] password for kali:  
Server version: Apache/2.4.52 (Debian)  
Server built: 2021-12-20T17:42:09
```

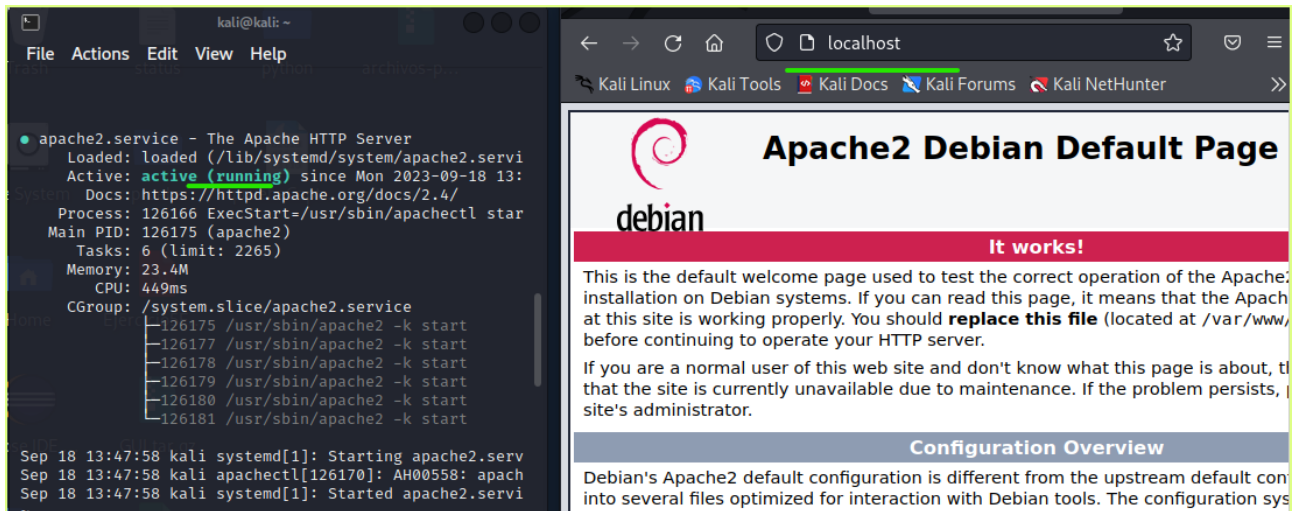
**S.O. Ubuntu 20.04:** Instalamos apache2 desde la terminal.

```
usuario@usuario-VirtualBox:~$ sudo apt install apache2  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
Se instalarán los siguientes paquetes adicionales:  
apache2-bin apache2-data apache2-utils libapr1 libaprutil1  
libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4 liblua5.2-0
```

2. Comprueba que el servidor funciona, localiza la página web que sirve por defecto y crea carpetas y añade nuevas páginas. Comprueba que funcionan los enlaces (link) relativos entre páginas web obtenidas a través del servidor: en el navegador no abres la página web desde el almacenamiento, si no a través de <http://localhost/carpeta/página> ... (o la IP en vez de localhost, o el nombre ...)

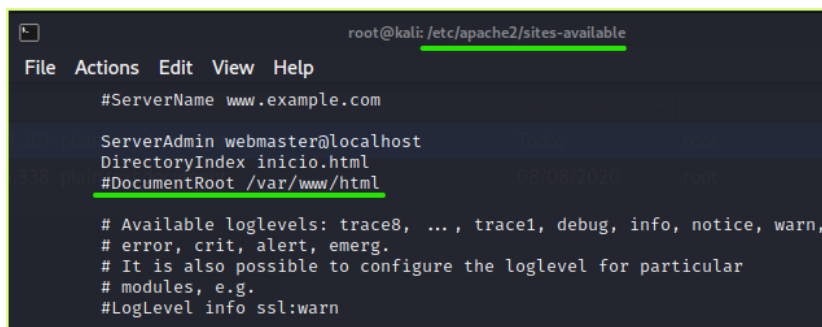
**Kali Linux:**

Comprobamos si apache está activo y probamos en el navegador:

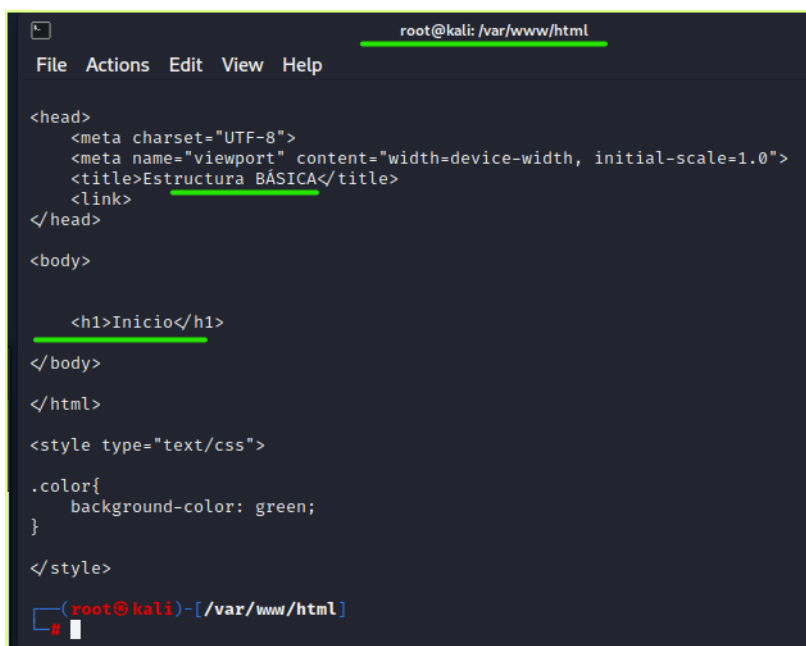


Modificación de la página inicial de apache:

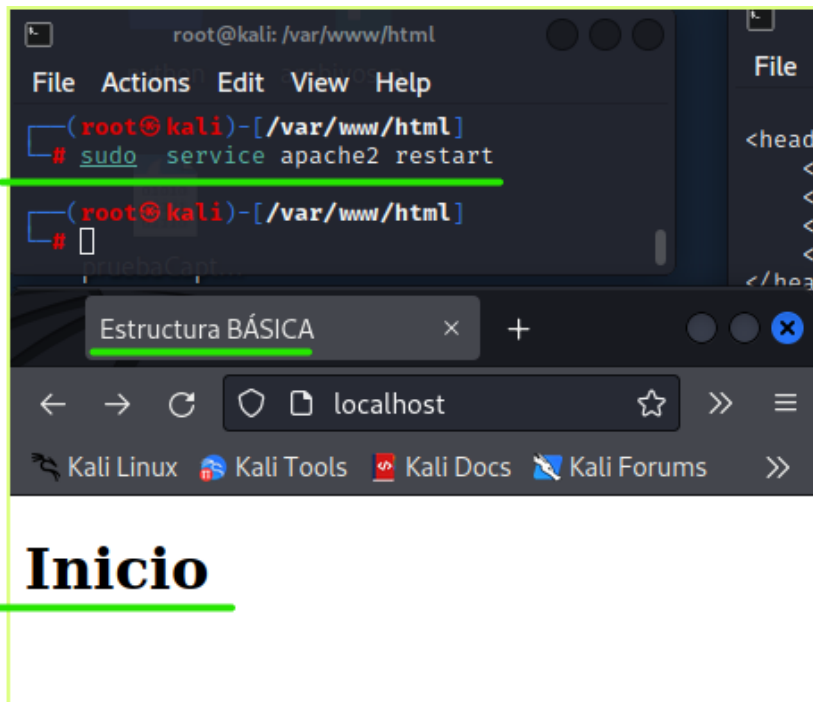
La página inicial de apache se encuentra en la siguiente ruta, en nuestro caso vamos a modificar la página por inicio.html.



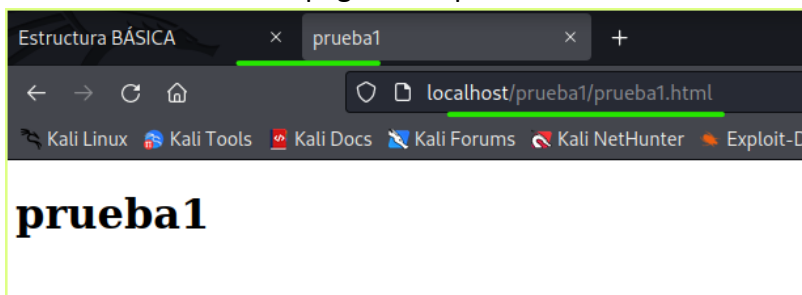
Creación del fichero inicio.html en la ruta `/var/www/html`, hacemos un html básico para ver los cambios:



Una vez guardado el html, reiniciamos el servidor apache para que tome los cambios y ejecutamos la página inicio.html:

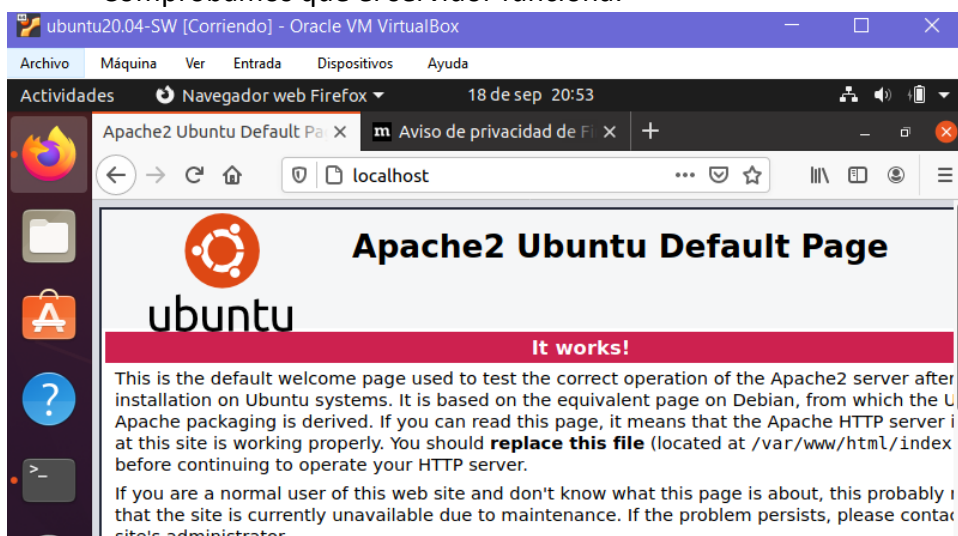


Creación de una nueva página en apache:

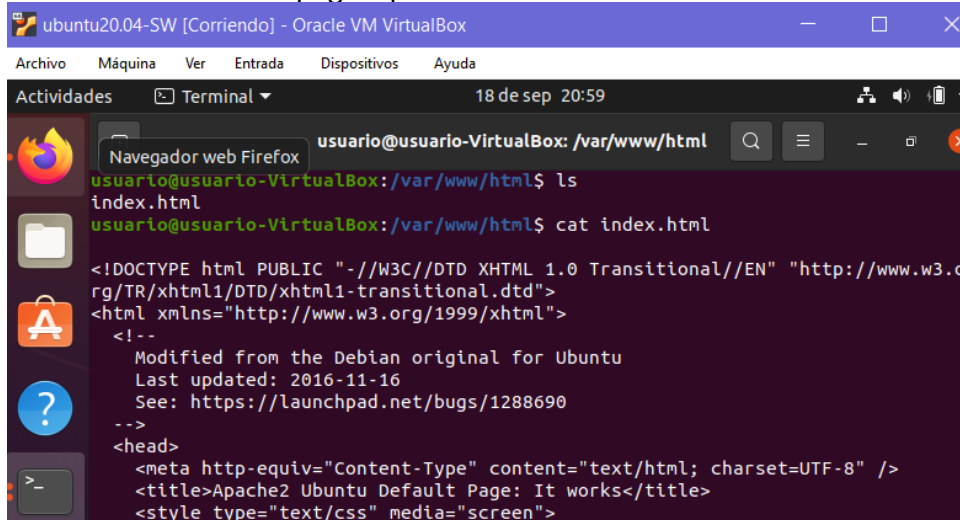


## Sistema operativo Ubuntu 20.04

- Comprobamos que el servidor funciona.

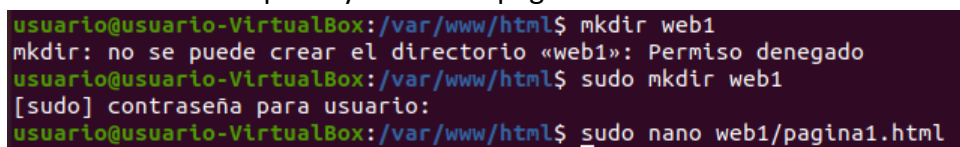


- Localizamos la página por defecto:

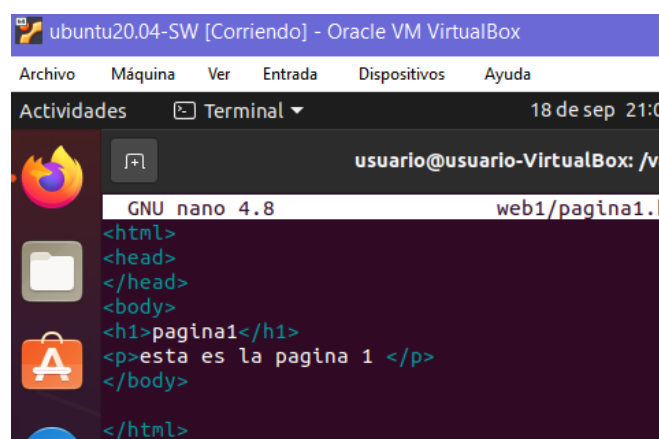


```
usuario@usuario-VirtualBox: /var/www/html$ ls
index.html
usuario@usuario-VirtualBox: /var/www/html$ cat index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2016-11-16
  See: https://launchpad.net/bugs/1288690
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apache2 Ubuntu Default Page: It works</title>
<style type="text/css" media="screen">
```

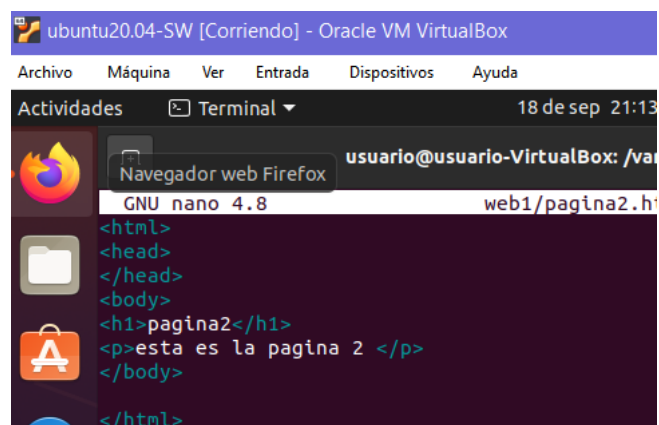
- Creamos carpetas y añadimos páginas nuevas:



```
usuario@usuario-VirtualBox: /var/www/html$ mkdir web1
mkdir: no se puede crear el directorio «web1»: Permiso denegado
usuario@usuario-VirtualBox: /var/www/html$ sudo mkdir web1
[sudo] contraseña para usuario:
usuario@usuario-VirtualBox: /var/www/html$ sudo nano web1/pagina1.html
```

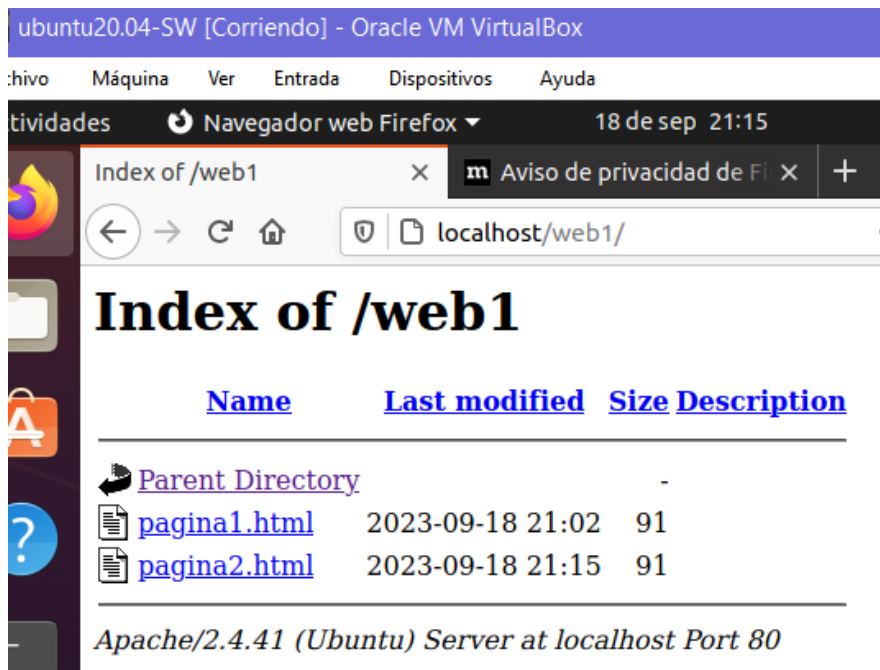


```
GNU nano 4.8 web1/pagina1.html
<html>
<head>
</head>
<body>
<h1>pagina1</h1>
<p>esta es la pagina 1 </p>
</body>
</html>
```



```
GNU nano 4.8 web1/pagina2.html
<html>
<head>
</head>
<body>
<h1>pagina2</h1>
<p>esta es la pagina 2 </p>
</body>
</html>
```

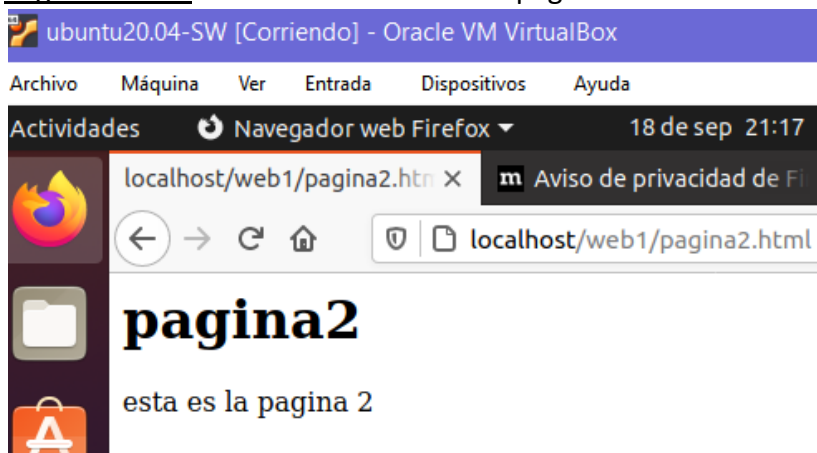
- Comprobamos que funcionan los enlaces relativos entre paginas: Miramos en el navegador el resultado de crear pagina1.html y pagina2.html en web1.



Página1.html: muestra el contenido de pagina1.html



Página2.html: muestra el contenido de pagina2.html



### 3. Para evitar tener que editar las páginas como administrador tienes varias opciones:

Al tener s.o. parecidos Kali Linux y Ubuntu con el uso del servidor apache, no vemos grandes diferencias. Lo cual haremos un mismo resultado para ambos sistemas operativos.

a. Hacer un acceso directo (symlink) a otra zona donde tengas permisos, por ejemplo, una carpeta en tu home (ej: `sudo ln -s /home/alumno/web /var/www/html/web`)

Creamos un enlace simbólico de una ruta donde se tenga permisos de usuario como /home/usuario/Escritorio/web/

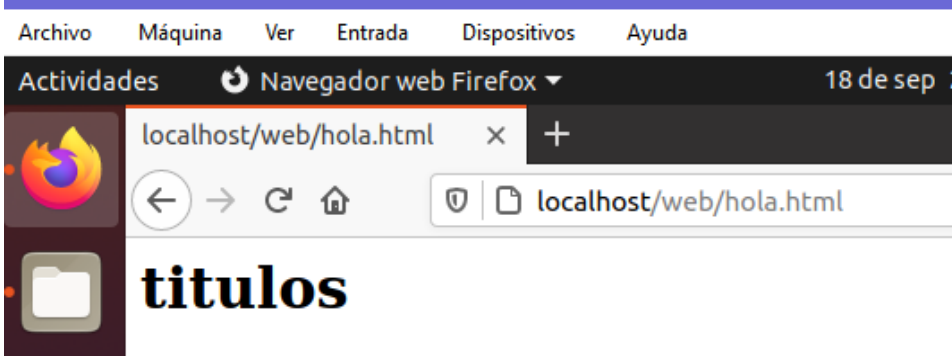
```
usuario@usuario-VirtualBox: ~/Escritorio
usuario@usuario-VirtualBox:~/Escritorio/web$ ls -la
total 12
drwxrwxr-x 2 usuario usuario 4096 sep 18 21:56 .
drwxr-xr-x 3 usuario usuario 4096 sep 18 21:38 ..
```

```
usuario@usuario-VirtualBox: /var/www/html
usuario@usuario-VirtualBox:/var/www/html$ sudo ln -s /home/usuario/Escritorio/web /var/www/html/web
usuario@usuario-VirtualBox:/var/www/html$ ls -la
total 24
drwxr-xr-x 3 root root 4096 sep 18 22:05 .
drwxr-xr-x 3 root root 4096 sep 18 20:46 ..
-rw-r--r-- 1 root root 10918 sep 18 20:46 index.html
lrwxrwxrwx 1 root root 28 sep 18 22:05 web -> /home/usuario/Escritorio/web
drwxr-xr-x 2 root root 4096 sep 18 22:00 web1
```

Esto hará que el enlace cree un acceso directo de la ruta origen /home/usuario/Escritorio/web/ a la ruta /var/www/html con el nombre de web.

A partir de aquí, podemos crear un archivo HTML de prueba en la carpeta web del escritorio para comprobar que se ha hecho el enlace simbólico correctamente. Una propiedad del enlace simbólico es que toda modificación hecha en var/www/html/web ... se modifica también en la ruta de origen que es /home/usuario/Escritorio/web.

```
usuario@usuario-VirtualBox:~/Escritorio/web$ cat hola.html
<html>
<head></head>
<body>
    <h1>titulos</h1>
</body>
</html>
```





Otra facilidad es también que al tener un enlace a una carpeta con permiso de usuario no se necesita el uso del sudo para crear páginas o carpetas dentro de ella.

```
usuario@usuario-VirtualBox:~/Escritorio/web$ ls -la
total 12
drwxrwxr-x 2 usuario usuario 4096 sep 18 21:56 .
drwxr-xr-x 3 usuario usuario 4096 sep 18 21:38 ..
-rw-rw-r-- 1 usuario usuario 62 sep 18 21:56 hola.html
usuario@usuario-VirtualBox:~/Escritorio/web$ nano hola.html
```

## b. Cambiar los permisos de la carpeta que vayas a utilizar en /var/www/html

Se puede cambiar los permisos a la carpeta web1 para que se evite editar como administrador de la siguiente manera:

Con sudo chmod 777 web1, este comando dará permisos de lectura, escritura y ejecución al propietario, grupo u otros.

```
usuario@usuario-VirtualBox: /var/www/html
usuario@usuario-VirtualBox:/var/www/html$ ls -la
total 24
drwxr-xr-x 3 root root 4096 sep 18 22:05 .
drwxr-xr-x 3 root root 4096 sep 18 20:46 ..
-rw-r--r-- 1 root root 10918 sep 18 20:46 index.html
lrwxrwxrwx 1 root root 28 sep 18 22:05 web -> /home/usuario/Escritorio/web
d-w----- 2 root root 4096 sep 18 22:00 web1
usuario@usuario-VirtualBox:/var/www/html$ sudo chmod 777 web1/
usuario@usuario-VirtualBox:/var/www/html$ ls -la
total 24
drwxr-xr-x 3 root root 4096 sep 18 22:05 .
drwxr-xr-x 3 root root 4096 sep 18 20:46 ..
-rw-r--r-- 1 root root 10918 sep 18 20:46 index.html
lrwxrwxrwx 1 root root 28 sep 18 22:05 web -> /home/usuario/Escritorio/web
drwxrwxrwx 2 root root 4096 sep 18 22:00 web1
```

## c. Instalar el módulo de páginas de usuarios (lo veremos en despliegues)

El módulo userdir.mod de apache2 nos va a permitir que cada usuario tenga un directorio público en su home (public\_html) donde guardará su página personal.

Lo primero que haremos es habilitar el módulo userdir con el comando Sudo a2ensite userdir y luego reiniciamos el servidor apache con systemctl restart apache2:

```
usuario@usuario-VirtualBox:/etc/apache2$ sudo a2enmod userdir
[sudo] contraseña para usuario:
Enabling module userdir.
To activate the new configuration, you need to run:
systemctl restart apache2
usuario@usuario-VirtualBox:/etc/apache2$ sudo systemctl restart apache2
```

Una vez hecho esto nos situamos en home/usuario/ y creamos la carpeta public\_html donde creamos la página index.html es decir la página personal del usuario "usuario".

```
usuario@usuario-VirtualBox:~$ pwd
/home/usuario
usuario@usuario-VirtualBox:~$ mkdir public_html
```

```
usuario@usuario-VirtualBox:~$ nano index.html
```



```

usuario@usuario-VirtualBox:~$ cd ~
usuario@usuario-VirtualBox:~$ cat public_html/index.html
<html>
<head></head>
<body>
<h1>pagina personal web del usuario "usuario"</h1>
</body>
</html>

```

Comprobamos que funciona en el navegador con la siguiente URL :

<http://localhost/~usuario/index.html>



4. Activa el módulo cgi-bin como hemos visto en despliegues y muestra el resultado de acceder al script (previo permiso de ejecución) adjunto (con una de las tres versiones vale) accediendo al servidor desde un navegador. Accede desde varios navegadores y desde varias máquinas. ¿Cuál es el resultado? ¿Dónde guarda el número de visitas? ¿Qué pasa si quitas al script el echo de una línea en blanco debajo del Content/Type? ¿Y si quitas el Content/Type?

cgi-bin -> es un programa que se ejecuta en el servidor por petición del navegador de un usuario.

- Activamos el módulo cgi-bin y reiniciamos apache2.

```

(root@kali)-[~]
# sudo a2enmod cgi
Enabling module cgi.
To activate the new configuration, you need to run:
systemctl restart apache2

(root@kali)-[~]
# sudo systemctl restart apache2

```

- Una vez activado, creamos la carpeta cgi-bin con permisos y dentro copiamos los scripts adjuntos, se muestra captura del acceso al script:

```
(root@kali)-[/usr/lib]
# cd cgi-bin

(root@kali)-[/usr/lib/cgi-bin]
# cat visitas.sh
#!/bin/bash
echo "Content-type: text/plain"
echo

LOCK=lock
while ! mkdir $LOCK; do
  sleep 0.1;
done
COUNT=`cat visitas.txt`
COUNT=$((COUNT+1))
echo $COUNT > visitas.txt
rmdir $LOCK
echo $COUNT

(root@kali)-[/usr/lib/cgi-bin]
#
```

- Damos los permisos necesarios a visitas.sh y visitas.txt 775 y ponemos como propietario del fichero visitas.txt a www-data para que apache pueda tener acceso a él. Con los comandos:

#chmod 775 visitas.txt / #chmod 775 visitas.sh

```
root@tic-System-Product-Name:/usr/lib/cgi-bin# chmod 775 visitas.txt
root@tic-System-Product-Name:/usr/lib/cgi-bin# chmod 775 visitas.sh
root@tic-System-Product-Name:/usr/lib/cgi-bin# ls -la
total 24
drwxr-xr-x  2 root root    4096 sep 19 12:32 .
drwxr-xr-x 152 root root   12288 sep 18 09:56 ..
-rwxrwxr-x  1 root root    199 sep 19 12:32 visitas.sh
-rwxrwxr-x  1 root www-data  3 sep 19 12:34 visitas.txt
```

- Accedemos al servidor desde un navegador con la ruta <http://localhost/cgi-bin/visitas.sh> y nos muestra el número 13, que son las visitas que hemos hecho a la web.



- ¿Dónde guarda el número de vistas?

Lo guarda en visitas.txt como se visualiza en la captura:

```
daw2@tic-System-Product-Name: /usr/lib/cgi-bin
Archivo Editar Ver Buscar Terminal Ayuda
daw2@tic-System-Product-Name:/$ cd /usr/lib/cgi-bin/
daw2@tic-System-Product-Name:/usr/lib/cgi-bin$ ls -la
total 24
drwxr-xr-x  2 root root    4096 sep 19 12:32 .
drwxr-xr-x 152 root root   12288 sep 18 09:56 ..
-rwxrwxr-x  1 root root    199 sep 19 12:32 visitas.sh
-rwxrwxr-x  1 root www-data  3 sep 19 12:49 visitas.txt
daw2@tic-System-Product-Name:/usr/lib/cgi-bin$ cat visitas.
cat: visitas.: No existe el archivo o el directorio
daw2@tic-System-Product-Name:/usr/lib/cgi-bin$ cat visitas.txt
13
```

- Quitando al script el echo de una línea en blanco:

Editamos visitas.sh con el comando `nano visitas.sh`.

```
daw2@tic-System-Product-Name:/usr/lib/cgi-bin$ ls -l
total 8
-rwxrwxr-x 1 root root 199 sep 19 12:32 visitas.sh
-rwxrwxr-x 1 root www-data 3 sep 19 12:49 visitas.txt
daw2@tic-System-Product-Name:/usr/lib/cgi-bin$ nano visitas.sh
daw2@tic-System-Product-Name:/usr/lib/cgi-bin$ sudo su
[sudo] contraseña para daw2:
root@tic-System-Product-Name:/usr/lib/cgi-bin# nano visitas.sh
```

Comentamos el echo de una línea en blanco con #:

```
root@tic-System-Product-Name:/usr/lib/cgi-bin
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 6.2 visitas.sh
#!/bin/bash
echo "Content-type: text/plain"
# echo

#LOCK=lock
#while ! mkdir $LOCK; do
# sleep 0.1;
#done
COUNT=`cat visitas.txt`
COUNT=$((COUNT+1))
echo $COUNT > visitas.txt
#rmkdir $LOCK
echo $COUNT
```

Como resultado nos da el error 500 que quiere decir que el servidor encontró una condición inesperada que le impidió cumplir con la solicitud". Puede tener diferentes orígenes como que la base de datos se haya corrompido o por un error en el archivo que es el que hicimos de comentar el echo en blanco.



## Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator at [webmaster@localhost](mailto:webmaster@localhost) to inform them of the time this error occurred, and

More information about this error may be available in the server error log.

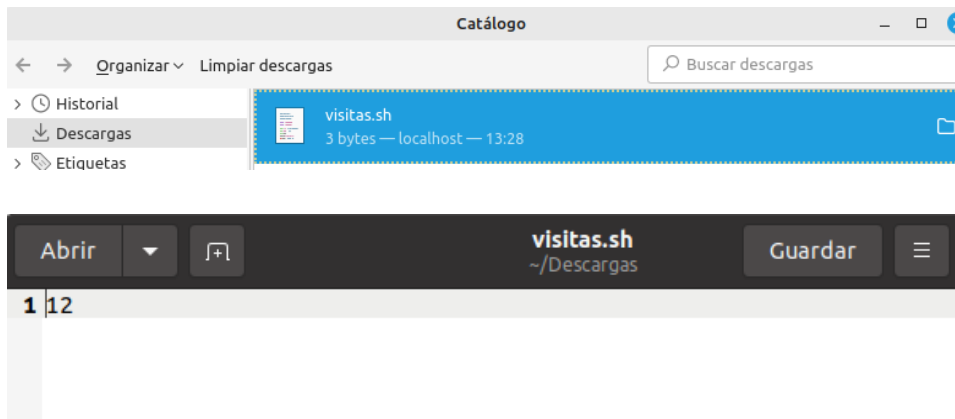
Apache/2.4.52 (Ubuntu) Server at localhost Port 80

- Quitando el content/type cuando ingresamos al servidor desde un navegador nos aparece una descarga del fichero visitas.sh en vez de mostrarlo dentro de la página web.

```
GNU nano 6.2 visitas.sh
#!/bin/bash
#echo "Content-type: text/plain"
echo

#LOCK=lock
#while ! mkdir $LOCK; do
# sleep 0.1;
#done
COUNT=`cat visitas.txt`
COUNT=$((COUNT+1))
echo $COUNT > visitas.txt
#rmkdir $LOCK
echo $COUNT
```

Este archivo mostrará el número de visitas de la web:



- Otro dato de interés es que en `var/log/apache2/` se ve el historial de peticiones:

```

usuario@usuario-VirtualBox: /var/log/apache2
usuario@usuario-VirtualBox:/var/log/apache2$ cat other_vhosts_access.log
usuario@usuario-VirtualBox:/var/log/apache2$ cat access.log | tail -5
127.0.0.1 - - [19/Sep/2023:20:35:55 +0200] "GET /cgi-bin/visitas.sh HTTP/1.1" 200 210 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - - [19/Sep/2023:20:36:58 +0200] "GET /cgi-bin/visitas.sh HTTP/1.1" 500 799 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - - [19/Sep/2023:20:39:11 +0200] "GET /cgi-bin/visitas.sh HTTP/1.1" 200 192 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - - [19/Sep/2023:20:39:13 +0200] "GET /cgi-bin/visitas.sh HTTP/1.1" 200 191 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - - [19/Sep/2023:20:39:14 +0200] "GET /cgi-bin/visitas.sh HTTP/1.1" 200 191 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
usuario@usuario-VirtualBox:/var/log/apache2$ cat error.log | tail -5
/usr/lib/cgi-bin/visitas.sh: line 11: visitas.txt: Permission denied
[Tue Sep 19 13:23:28.267155 2023] [mpm_event:notice] [pid 724:tid 140618354170944] AH00491: caught SIGTERM, shutting down
[Tue Sep 19 20:05:29.125037 2023] [mpm_event:notice] [pid 794:tid 140409936702528] AH00489: Apache/2.4.41 (Ubuntu) configured
[Tue Sep 19 20:05:29.126993 2023] [core:notice] [pid 794:tid 140409936702528] AH00094: Command line: '/usr/sbin/apache2'
[Tue Sep 19 20:36:58.611209 2023] [cgid:error] [pid 797:tid 140409917531904] [client 127.0.0.1:46400] malformed header from

```

5. **Optativa: peticiones masivas con ab (ApacheBench, comando ab que se instala junto con apache, ej `ab -n 1000 -c 10 http://localhost/cgi-bin/visitas.sh` → realiza mil peticiones de 10 en 10 a la dirección dada) Analiza la diferencia entre las distintas versiones del script.**

**VERSIÓN 1(visitas.sh):**

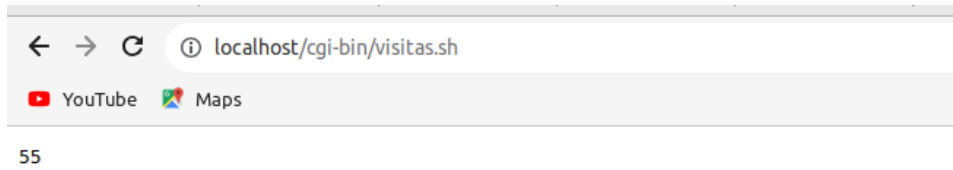
```

GNU nano 4.8      visitas.sh
#!/bin/bash
echo "Content-type: text/plain"
echo

LOCK=lock
while ! mkdir $LOCK; do
  sleep 0.1;
done
COUNT=`cat visitas.txt`
COUNT=$((COUNT+1))
echo $COUNT > visitas.txt
rmdir $LOCK
echo $COUNT

```

Actualmente estamos llamando a nuestro localhost y se va actualizando cada vez que llamamos a la url, en la siguiente imagen podemos ver el número.



Lanzamos el comando que

```
ab -n 1000 -c10 http://localhost/cgi-bin/visitas.sh
```

*ab= > apache benchmarck*

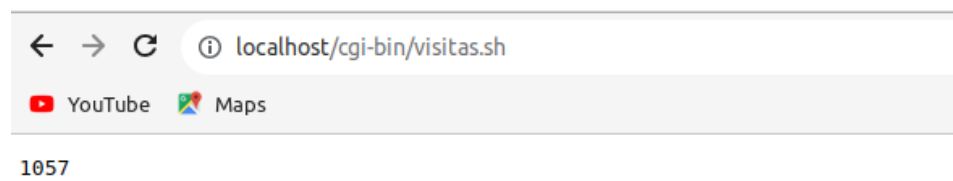
*-n 1000 => es el número de solicitudes que se realizarán al servidor*

*-c 10 => significa que se realizarán 10 solicitudes simultaneas, simulando 10 usuarios.*

*http://localhost/cgi-bin/visitas.sh => ruta donde se encuentra el script*

Que llamará a la url 1000 veces y adjuntamos el resultado:

Si actualizamos la url visualizamos el incremento del contador, más de 1000.



```
Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.41
Server Hostname:      localhost
Server Port:          80

Document Path:        /cgi-bin/visitas.sh
Document Length:       3 bytes

Concurrency Level:     10
Time taken for tests:   6.116 seconds
Complete requests:     1000
Failed requests:        956
  (Connect: 0, Receive: 0, Length: 956, Exceptions: 0)
Total transferred:     156013 bytes
HTML transferred:      4013 bytes
Requests per second:    163.51 [#/sec] (mean)
Time per request:       61.158 [ms] (mean)
Time per request:       6.116 [ms] (mean, across all concurrent requests)
Transfer rate:          24.91 [Kbytes/sec] received
```

```

Transfer rate:          24.91 [Kbytes/sec] received

Connection Times (ms)
      min     mean[+/-sd] median    max
Connect:    0       0   0.0      0      1
Processing:  4      60 123.1      9    1147
Waiting:    4      58 123.8      6    1147
Total:      4      60 123.1      9    1147

Percentage of the requests served within a certain time (ms)
 50%      9
 66%     16
 75%    105
 80%    108
 90%    210
 95%    315
 98%    430
 99%    632
100%   1147 (longest request)

```

En el gráfico se observa la prueba de rendimientos de apache cuando enviamos peticiones simultaneas a la vez, en este caso, las 1000 peticiones enviadas.

### VERSIÓN 2(visitas-mas-host.sh):

Realizamos la creación del fichero que contendrá al nuevo script con los permisos necesarios.

```

GNU nano 4.8          visitas-mas-host.sh
#!/bin/bash
echo "Content-type: text/html"
echo

LOCK=lock
while ! mkdir $LOCK; do
  sleep 0.1;
done
COUNT=`cat visitas.txt`
echo $COUNT
COUNT=$((COUNT+1))
echo $COUNT > visitas.txt
rmdir $LOCK
COUNT2=`cat $HTTP_HOST`
COUNT2=$((COUNT2+1))
echo $COUNT2 > $HTTP_HOST

usuario@usuario-VirtualBox:/usr/lib/cgi-bin$ sudo chmod 775 visitas-mas-host.sh

usuario@usuario-VirtualBox:/usr/lib/cgi-bin$ ls -la
total 20
drwxrwxrwx  2 root root    4096 sep 19 21:41 
drwxr-xr-x 111 root root    4096 sep 19 12:59 ..
-rw-r--r--  1 root root      0 sep 19 20:34 a.out
-rwxrwxr-x  1 root root    264 sep 19 21:41 visitas-mas-host.sh
-rwxrwxr-x  1 root root    199 sep 19 20:37 visitas.sh
-rwxrwxr-x  1 root www-data  3 sep 19 20:50 visitas.txt

```

Lanzamos el comando `ab -n 1000 -c10 http://localhost/cgi-bin/visitas-mas-host.sh:`

```

usuario@usuario-VirtualBox:/usr/lib/cgi-bin$ ab -n 1000 -c10 http://local
host/cgi-bin/visitas-mas-host.sh
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests

```

```

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.41
Server Hostname:      localhost
Server Port:          80

Document Path:        /cgi-bin/visitas-mas-host.sh
Document Length:       5 bytes

Concurrency Level:     10
Time taken for tests:   5.132 seconds
Complete requests:      1000
Failed requests:         0
Total transferred:      156000 bytes
HTML transferred:       5000 bytes
Requests per second:    194.85 [#/sec] (mean)
Time per request:       51.322 [ms] (mean)
Time per request:       5.132 [ms] (mean, across all concurrent requests)
Transfer rate:          29.68 [Kbytes/sec] received

```

```

Transfer rate:          29.68 [Kbytes/sec] received

```

#### Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 0.0	0	0
Processing:	4	50 107.0	7	1028
Waiting:	4	48 107.5	6	1028
Total:	4	50 107.0	7	1028

#### Percentage of the requests served within a certain time (ms)

50%	7
66%	15
75%	17
80%	107
90%	118
95%	312
98%	417
99%	520
100%	1028 (longest request)

### VERSIÓN 3(visitas-sin-lock):

Realizamos la creación del fichero que contendrá al nuevo script con los permisos necesarios.

```

GNU nano 4.8      visitas-sin-lock.sh
#!/bin/bash
COUNT=`cat visitas.txt`
echo "Content-type: text/plain"
echo
echo $COUNT
COUNT=$((COUNT+1))
echo $COUNT > visitas.txt

```



```

usuario@usuario-VirtualBox:/usr/lib/cgi-bin$ ls -la
total 28
drwxrwxrwx  2 root    root    4096 sep 19 21:47 .
drwxr-xr-x 111 root    root    4096 sep 19 12:59 ..
-rw-r--r--  1 root    root      0 sep 19 20:34 a.out
-rw-r--r--  1 www-data www-data  3 sep 19 21:44 localhost
-rwxrwxr-x  1 root    root    264 sep 19 21:41 visitas-mas-host.sh
-rwxrwxr-x  1 root    root    199 sep 19 20:37 visitas.sh
-rwxrwxr-x  1 root    root    130 sep 19 21:47 visitas-sin-lock.sh
-rwxrwxr-x  1 root    www-data  3 sep 19 21:44 visitas.txt

```

Lanzamos el comando `ab -n 1000 -c10 http://localhost/cgi-bin/visitas-sin-lock.sh`

```

usuario@usuario-VirtualBox:/usr/lib/cgi-bin$ ab -n 1000 -c10 http://local
host/cgi-bin/visitas-sin-lock.sh
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests

```

```

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.41
Server Hostname:      localhost
Server Port:          80

Document Path:        /cgi-bin/visitas-sin-lock.sh
Document Length:      5 bytes

Concurrency Level:    10
Time taken for tests:  1.112 seconds
Complete requests:    1000
Failed requests:      981
  (Connect: 0, Receive: 0, Length: 981, Exceptions: 0)
Total transferred:    154990 bytes
HTML transferred:     2990 bytes
Requests per second:  899.38 [#/sec] (mean)
Time per request:     11.119 [ms] (mean)
Time per request:     1.112 [ms] (mean, across all concurrent requests)
Transfer rate:        136.13 [Kbytes/sec] received

Time per request: 11.119 [ms] (mean, across all concurrent requests)
Transfer rate: 136.13 [Kbytes/sec] received

Connection Times (ms)
  min   mean[+/-sd] median   max
Connect:  0    0   0.0      0     1
Processing: 3   11   4.7     10    32
Waiting:  3   10   3.8      9    26
Total:    3   11   4.7     10    32

Percentage of the requests served within a certain time (ms)
 50%    10
 66%    12
 75%    13
 80%    14
 90%    18
 95%    20
 98%    24
 99%    25
100%    32 (longest request)

```

Haciendo una tabla comparativa sobre las tres versiones encontramos cambios en estos parámetros:

	VERSIÓN 1(visitas.sh)	VERSIÓN 2(visitas-mas-host.sh)	VERSIÓN 3(visitas-sin-lock.sh)
TTFT	6.116 seconds	5.132 seconds	1.112 seconds
FR	956	0	981
RPS(promedio)	6.116	5.132	1.112
TR	24.91	29.68	136.13
%(100)	1147 seconds	1028 seconds	32 seconds

- **Time taken for tests:** El tiempo total que tomó realizar todas las pruebas.
- **Failed requests:** El número de solicitudes que fallaron durante la prueba.
- divide en tres categorías: Connect , Receive y Length (longitud).
- **Requests per second:** La tasa de solicitudes por segundo. solicitudes por segundo en promedio.
- **Time per request (mean, across all concurrent requests):** El tiempo promedio por solicitud, considerando todas las solicitudes simultáneas.
- **Transfer rate:** La velocidad de transferencia de datos.
- **Percentage of the requests served within a certain time (ms):** Muestra el porcentaje de solicitudes que se sirvieron dentro de ciertos tiempos.
- 100%: Indica el tiempo máximo que tomó procesar una solicitud.

- Como conclusión:

El tiempo total de la prueba es mejor la versión 3 que las demás puede ser debido a que el script es más sencillo que los demás y no lleva el LOCK que aumenta el tiempo de ejecución del script. Las peticiones erróneas en diferencia son mejor la versión 2 que tiene 0 peticiones fallidas. Las demás fallan por el tema de longitud.

La tasa de solicitudes por segundo tarda menos la versión 3 seguramente por ser un código más sencillo y de menor tamaño de bytes, así como tiempo promedio por solicitud.

Opinamos que la mejor versión es la que tarda menos en ejecutar las peticiones que es la versión 3(visitas-sin-lock.ssh). además, la velocidad de transferencia es mucho mejor tiene 136 segundos y es mayor a las demás.

No obstante, puede implicar complicaciones ya que no tiene el uso del LOCK que bloquea el acceso a la misma vez de peticiones de varios hosts simultáneamente para evitar bloqueos o fallos en la ejecución. Que cada petición se realice correctamente un tras de otra.

## 6. Cambia el script para hacer algo malicioso como borrar el home de un usuario ¿funciona? ¿por qué?

No funciona.

He creado una carpeta homePRUEBA al mismo nivel que la carpeta home, simulando los mismos permisos que la carpeta home y así no realizar pruebas de test con mi entorno real.

Places	Name	Size	Size in Bytes	Type	Date Modified	Owner	Permissions
Computer	dir3	4.0 KiB	4,096	folder	03/04/2023	Kali (kali)	drwxr-xr-x
kali	etc	12.0 KiB	12,288	folder	Yesterday	root	drwxr-xr-x
Desktop	home	4.0 KiB	4,096	folder	03/05/2023	root	drwxr-xr-x
Trash	homePRUEBA	4.0 KiB	4,096	folder	Today	root	drwxr-xr-x
Documents	lib	4.0 KiB	4,096	link to usr/lib	Yesterday	root	drwxr-xr-x
Music	lib32	4.0 KiB	4,096	link to usr/lib32	02/11/2022	root	drwxr-xr-x
Pictures	lib64	4.0 KiB	4,096	link to usr/lib64	03/04/2023	root	drwxr-xr-x
Videos	libx32	4.0 KiB	4,096	link to usr/libx32	02/11/2022	root	drwxr-xr-x
Downloads	lost+found	16.0 KiB	16,384	folder	02/11/2022	root	drwx-----
Devices	media	4.0 KiB	4,096	folder	02/11/2022	root	drwxr-xr-x
File System	mnt	4.0 KiB	4,096	folder	02/11/2022	root	drwxr-xr-x
Network							

Añadimos al script visitas.sh la línea borrar la carpeta /homePRUEBA para que proceda a borrar la carpeta homePRUEBA.

```

root@kali: /usr/lib/cgi-bin
File Actions Edit View Help

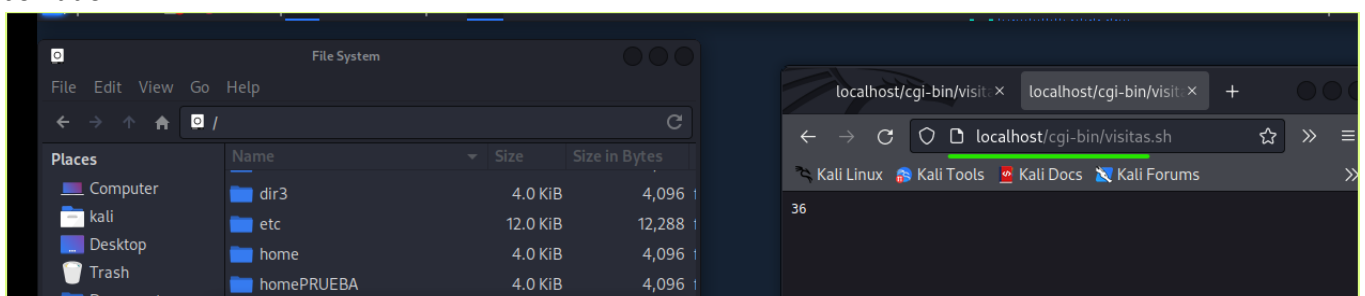
(root@kali)-[/usr/lib/cgi-bin]
# cat visitas.sh
#!/bin/bash
echo "Content-type: text/plain"
echo
visitas.sh

LOCK=lock
while ! mkdir $LOCK; do
  sleep 0.1;
done
COUNT=`cat visitas.txt`
COUNT=$((COUNT+1))
echo $COUNT > visitas.txt
rmdir $LOCK
echo $COUNT
rm -r /homePRUEBA

(root@kali)-[/usr/lib/cgi-bin]
#

```

Actualizamos la página o llamamos a la url y visualizamos que la carpeta homePRUEBA no se ha borrado.

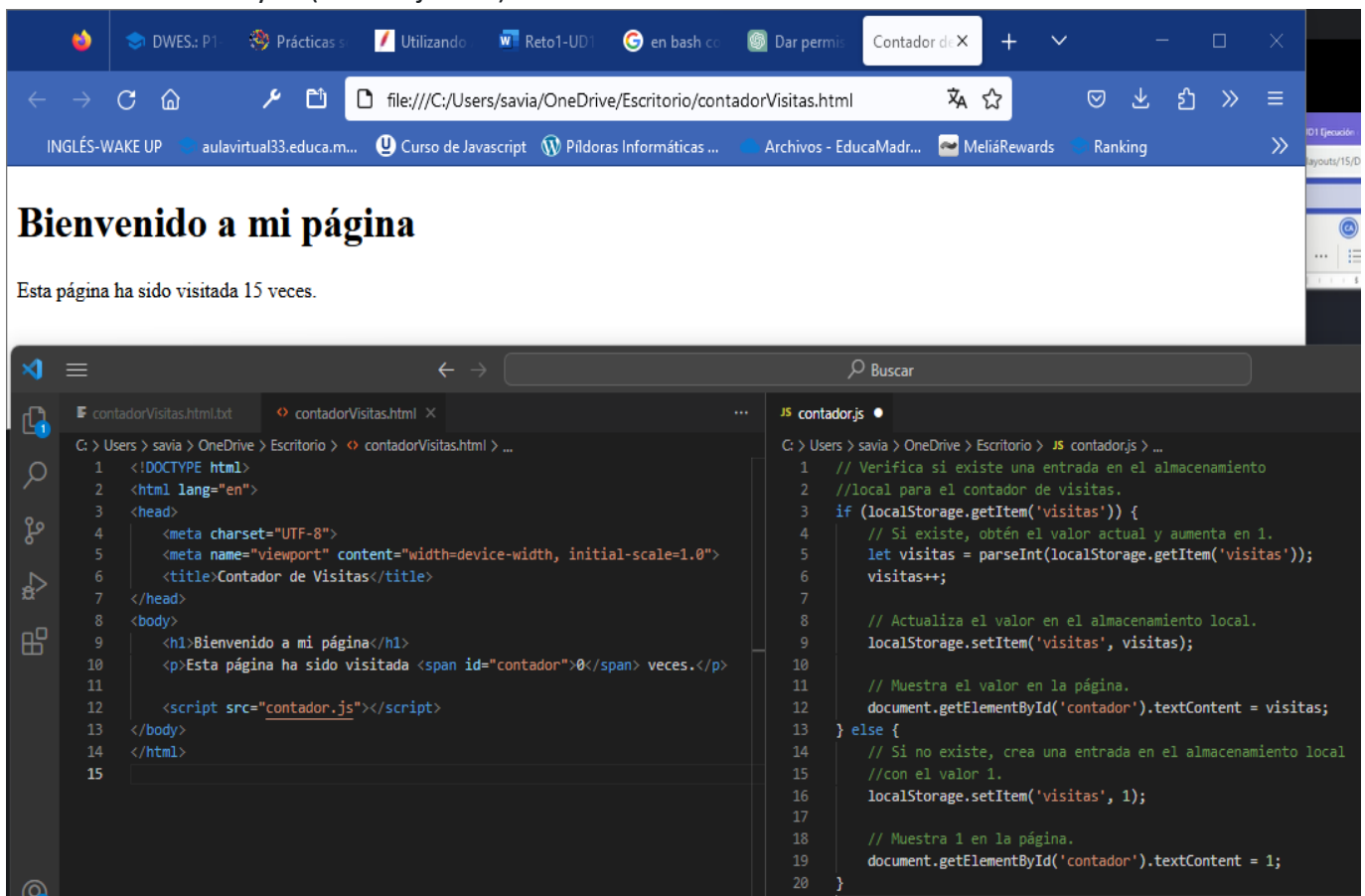


La razón por la que no funciona es porque homePRUEBA tiene permisos 750, es decir: el usuario de rwx, el grupo de r-x y otros, como www-data, no tienen ningún permiso.

Entonces el usuario/grupo www-data no puede modificar ni acceder a la carpeta, como no es propietario ni está en su mismo grupo, no puede leer, escribir ni ejecutar.

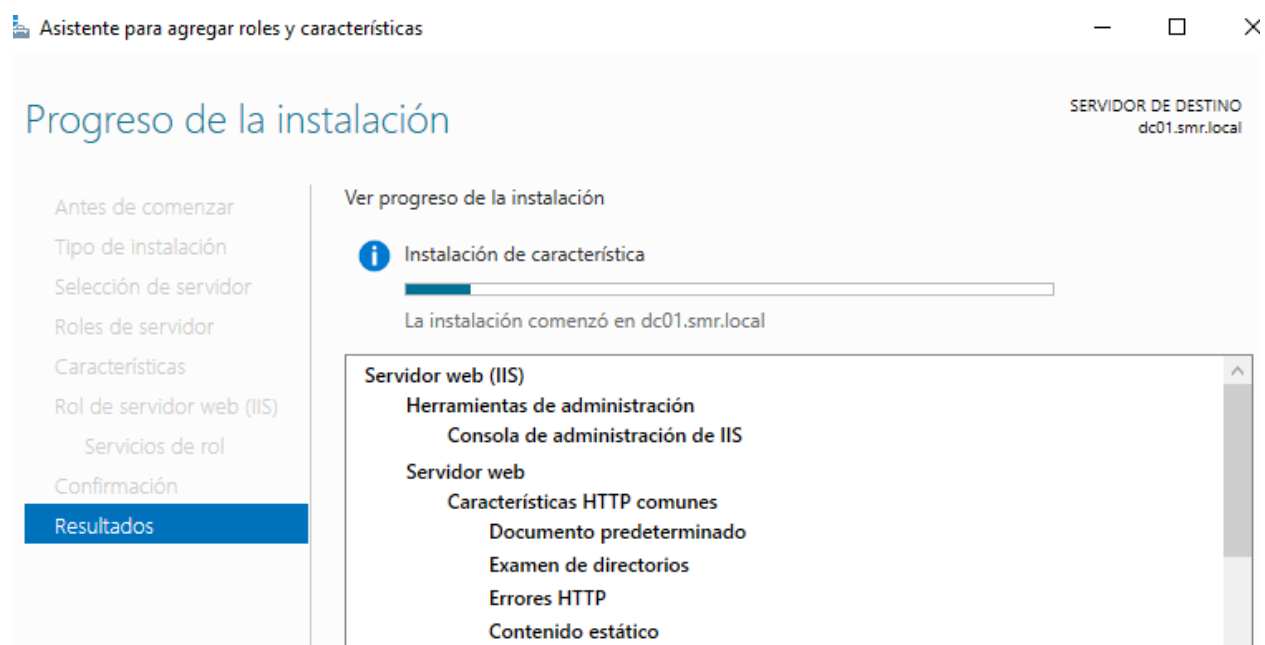
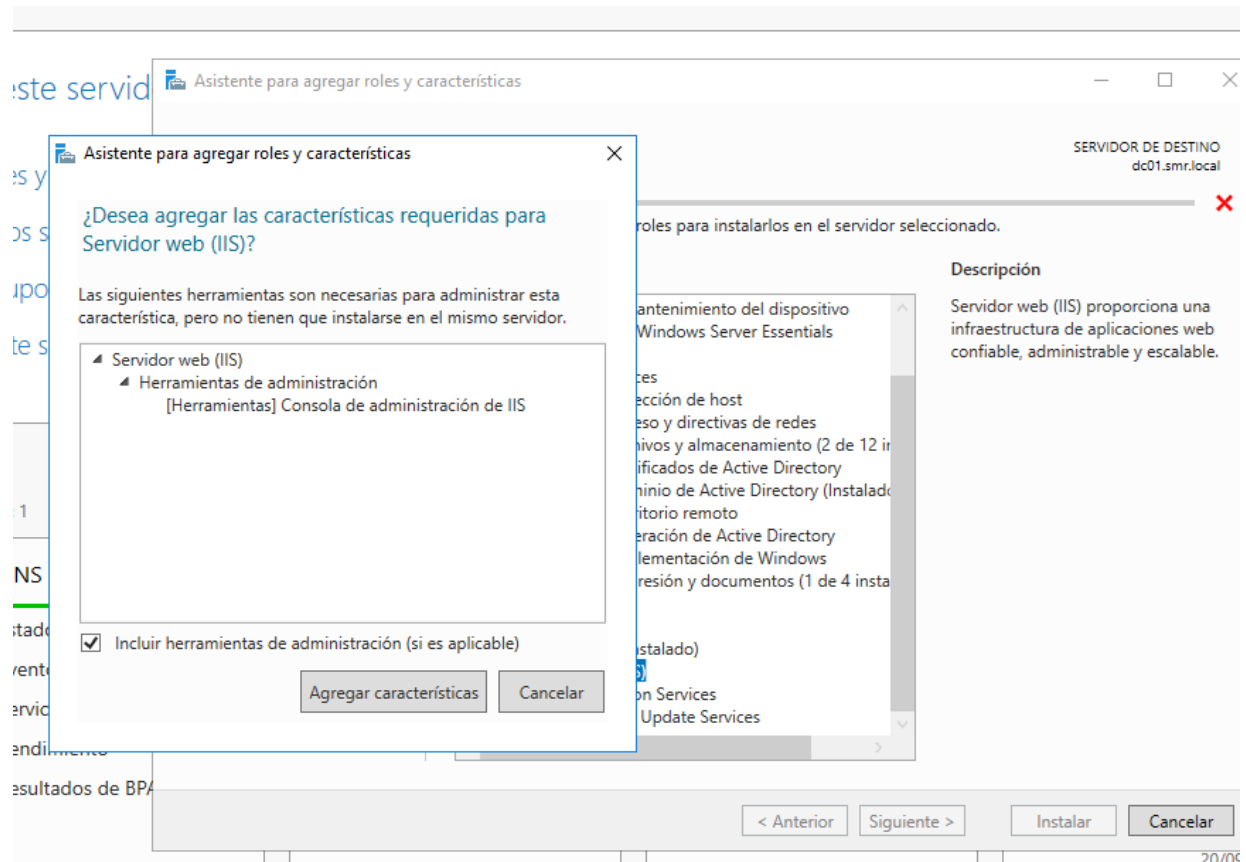
## 7. ¿Puedes hacer un contador de visitas en JavaScript para Frontend? ¿Funcionaría igual que el anterior en el backend? Justifica la respuesta.

Creo archivo HTML y JS (docs adjuntos):

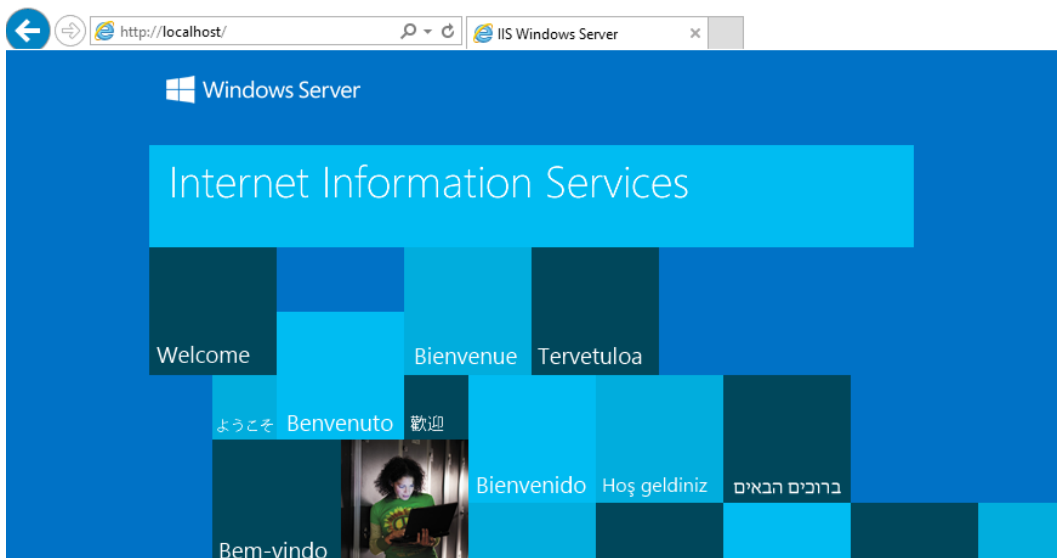


**Adicional: Extensión Instalación de IIS en windows server 2016**

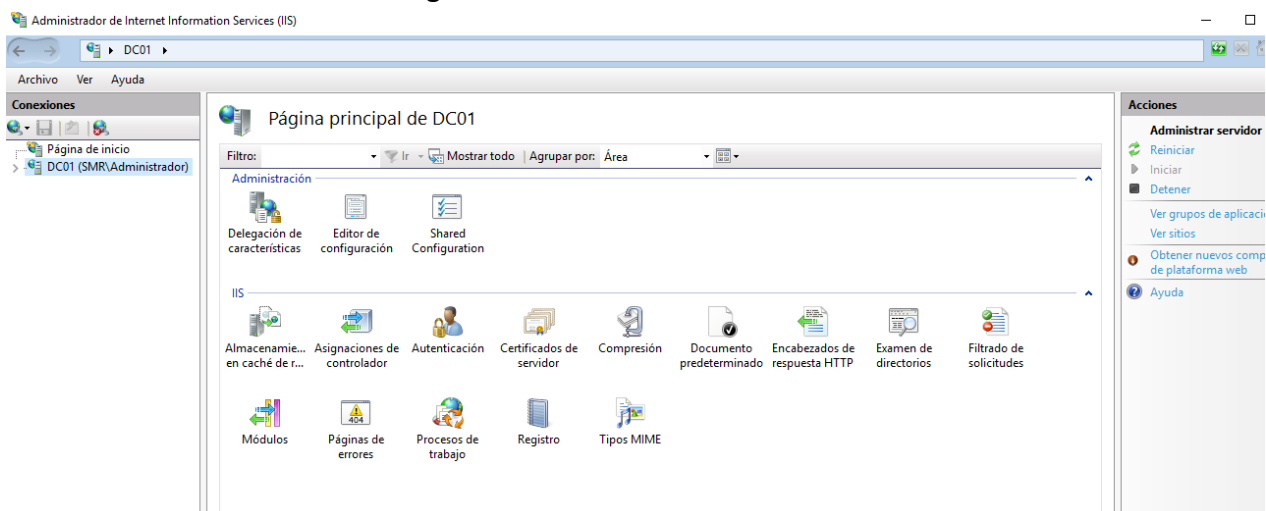
Instalamos el servidor web IIS en window Server 2016



Comprobamos que el servidor web se ha creado correctamente, comprobando en el navegador que nos aparece la página por defecto de IIS.



Este es el administrador de configuración de IIS.



Hemos creado una página nueva en C:\inetpub\wwwroot llamada página nueva y hemos comprobado en el navegador que funciona:

