

Trabajo Práctico

TPL 5 - World Wide Web - HTTP

Fecha de entrega: 26/05/2021

Franco, Juan Martín 149.615

juanmartin_franco@hotmail.com

1. Describa someramente el protocolo HTTP. Indique modo de operación y primitivas básicas.

El protocolo HTTP (Hypertext Transfer Protocol, Protocolo de transferencia de Hipertexto) es uno de los tantos protocolos de aplicación que corren sobre el stack TCP/IP.

Éste protocolo es de tipo Cliente-Servidor, lo que significa que hay una interacción, donde el cliente es un componente que realiza peticiones de servicio a otro componente. Este último componente que recibe y satisface dichas peticiones es el servidor. Ambos componentes están conectados mediante algún tipo de red.

El User Agent más común de HTTP es el Web Browser.

Del lado del servidor el puerto bien conocido del protocolo de aplicación HTTP es el puerto 80.

Modo de operación:

Básicamente lo que hace HTTP es intercambiar recursos distribuidos en distintos servidores alrededor del mundo.

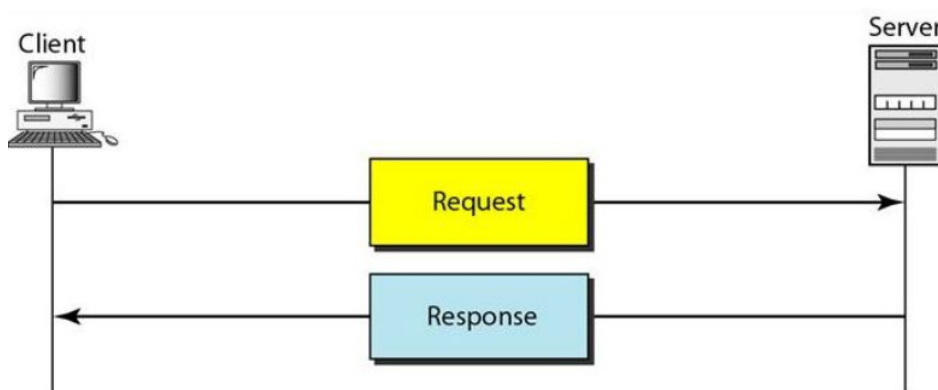
El equipo cliente, de alguna forma tiene una dirección, y hace una petición a un servidor web en el puerto 80.

El web server atiende esta petición (se abre una conexión TCP), es decir, la interpreta, por lo que va a buscar el archivo de la petición, lo abre, lo lee y se lo devuelve (por la misma conexión TCP abierta con anterioridad) al user agent.

El archivo que fue devuelto al user agent, cuenta con alguna estructura adicional que indica ciertos datos sobre este recurso y sobre la transacción.

HTTP solo se encarga de la transferencia, por lo que no tiene responsabilidad por ejemplo en cuestiones relacionadas a como se muestra en pantalla dichos recursos, como se almacena en servidores, etc.

Imagen representativa de una transacción HTTP:



Primitivas básicas:

Método	Acción
GET	Solicitud de un documento a un servidor.
HEAD	Solicitud de información de un documento (encabezados), sin transferir el cuerpo de la entidad.
POST	Envío de alguna información del cliente al servidor.
PUT	Es una solicitud para aceptar la entidad adjunta por parte del servidor. Esto puede significar un nuevo recurso o la sustitución de contenido existente en una URL.
DELETE	Elimina un recurso del servidor.
OPTIONS	Solicitud de información acerca de las opciones disponibles.

2. ¿Qué es HTML? ¿Qué especifica? Ejemplifique.

Los recursos nombrados con anterioridad, en general, están especificados en un lenguaje llamado HTML (HyperText Markup Language, Lenguaje de marcado de Hipertexto).

Este lenguaje de marcado es un estándar en lo que respecta a la creación de páginas web y para que sus datos sean formateados de forma que se pueda llevar a cabo su interpretación por los navegadores.

Una página web contiene solamente texto, por lo que la responsabilidad de interpretar dicho texto y unir los elementos que pueden llegar a estar involucrados (como imágenes, videos, etc.) recae sobre el navegador web.

HTML especifica básicamente etiquetas, y cada una de ellas es un nombre seguido de una lista de atributos (la cual es opcional). Las etiquetas se encierran entre < y >, y los atributos se definen en su interior con la sintaxis nombreAtributo = 'Valor'.

Ejemplo de la estructura de una página web:

```
<!DOCTYPE HTML>

<html>

  <head>

    <title>Título de mi página</title>

  </head>

  <body>

    <h1>Título visible en el cuerpo de la página</h1>

  </body>

</html>
```

3. Utilizando la herramienta nc , conéctese a la dirección y al puerto del servidor web www.unlu.edu.ar y lleve a cabo las siguientes pruebas utilizando primitivas del protocolo HTTP. Guarde las respuestas obtenidas.

a. Petición por protocolo HTTP versión 1.0

```
$ nc -v -C www.unlu.edu.ar 80 (enter)
GET / HTTP/1.0 (enter) (enter)
```

```
juanma@ubuntu:~$ nc -v -C www.unlu.edu.ar 80
Connection to www.unlu.edu.ar 80 port [tcp/http] succeeded!
GET / HTTP/1.0

HTTP/1.1 200 OK
```

b. Petición por protocolo HTTP versión 1.1

```
$ nc -v -C www.unlu.edu.ar 80 (enter)
GET / HTTP/1.1 (enter)
Host: www.unlu.edu.ar (enter) (enter)
```

```
juanma@ubuntu:~$ nc -v -C www.unlu.edu.ar 80
Connection to www.unlu.edu.ar 80 port [tcp/http] succeeded!
GET / HTTP/1.1
Host: www.unlu.edu.ar

HTTP/1.1 200 OK
```

c. Petición HTTP. Copie el texto de la petición (indicada bajo la línea) y péguelo una vez establecida la conexión con nc . Finalice la petición pulsando tres veces la tecla Enter.

```
$ nc -v -C www.unlu.edu.ar 80
GET / HTTP/1.1
Host: www.labredes.unlu.edu.ar
Connection: keep-alive
```

```
juanma@ubuntu:~$ nc -v -C www.unlu.edu.ar 80
Connection to www.unlu.edu.ar 80 port [tcp/http] succeeded!
GET / HTTP/1.1
Host: www.labredes.unlu.edu.ar
Connection: keep-alive

HTTP/1.1 200 OK
```

d. Petición HTTP. Copie el texto de la petición (indicada bajo la línea) y péguelo una vez establecida la conexión con nc . Finalice la petición pulsando tres veces la tecla Enter.

```
$ nc -v -C www.unlu.edu.ar 80
GET / HTTP/1.1
Host: www.labredes.unlu.edu.ar
Connection: close
```

```
juanma@ubuntu:~$ nc -v -C www.unlu.edu.ar 80
Connection to www.unlu.edu.ar 80 port [tcp/http] succeeded!
GET / HTTP/1.1
Host: www.labredes.unlu.edu.ar
Connection: close

HTTP/1.1 200 OK
```

Responda:

- a. **¿Qué códigos numéricos de respuesta HTTP devuelve el servidor web para cada petición? ¿Qué significan según la RFC?**

En la petición a) se obtuvo un código de estado 200 OK.

En la petición b) se obtuvo un código de estado 200 OK.

En la petición c) se obtuvo un código de estado 200 OK.

En la petición d) se obtuvo un código de estado 200 OK.

Según la RFC, este código de estado indica que la solicitud del cliente fue recibida con éxito, entendida y aceptada.

- b. **¿Cuáles son los otros encabezados devueltos y qué contenido es transferido en cada caso?**

a) Petición por protocolo HTTP versión 1.0

```
Date: Tue, 25 May 2021 21:03:29 GMT
Server: Apache
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

El contenido transferido es el código HTML de la página de la UNLu.

b) Petición por protocolo HTTP versión 1.1

```
Date: Tue, 25 May 2021 21:09:32 GMT
Server: Apache
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
```

El contenido transferido es el código HTML de la página de la UNLu.

c) Petición HTTP

```
Date: Tue, 25 May 2021 21:14:30 GMT
Server: Apache
X-Content-Type-Options: nosniff
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Cache-Control: no-cache, must-revalidate
X-Content-Type-Options: nosniff
Content-Language: es
X-Frame-Options: SAMEORIGIN
X-Generator: Drupal 7 (http://drupal.org)
Vary: Accept-Encoding
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

El contenido transferido es el código HTML de la página de Teleinformática y Redes.

d) Petición HTTP

```
Date: Tue, 25 May 2021 21:20:23 GMT
Server: Apache
X-Content-Type-Options: nosniff
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Cache-Control: no-cache, must-revalidate
X-Content-Type-Options: nosniff
Content-Language: es
X-Frame-Options: SAMEORIGIN
X-Generator: Drupal 7 (http://drupal.org)
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

El contenido transferido es el código HTML de la página de Teleinformática y Redes.

c. **¿Qué diferencia nota respecto a la duración de la conexión contra el servidor en los dos últimos casos?**

La diferencia que noto respecto a la duración de la conexión contra el servidor en los últimos 2 casos radica en que en la petición c) el servidor luego de responder la solicitud deja abierta la conexión a la espera de mas peticiones, en cambio en la solicitud d) una vez que se da respuesta a la petición se cierra la conexión.

4. **Realice 3 capturas de peticiones HTTP al servidor web www.unlu.edu.ar. Para la primer y segunda captura utilice 2 navegadores gráficos distintos (ej.: Firefox, Icesweasel, Chrome, Chromium, Konqueror, Epiphany, Explorer, Safari, etc.), y para la tercer captura use la herramienta de transferencias curl (<https://curl.haxx.se/>) o wget (<http://www.gnu.org/software/wget/>).**

a. **¿Qué encabezados envía cada cliente en la petición?**

Encabezados de Firefox:

```
GET / HTTP/1.1
Host: www.unlu.edu.ar
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1
```

Cache-Control: max-age=0

Encabezados de Chromium:

```
GET / HTTP/1.1
Host: www.unlu.edu.ar
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
dnt: 1
sec-gpc: 1
```

Encabezados de Curl:

Para realizar esta captura, utilicé el siguiente comando:

```
curl -v www.unlu.edu.ar
```

Encabezados:

```
GET / HTTP/1.1
Host: www.unlu.edu.ar
User-Agent: curl/7.68.0
Accept: */*
```

b. Comente las características de la información en tránsito con respecto a la confidencialidad.

En HTTP se puede afirmar que el contenido viaja sin ningún tipo de confidencialidad (texto plano), por lo que es un protocolo no seguro para las aplicaciones que requieran un login o el envío de datos que puedan llegar a ser confidenciales como por ej. datos de una cuenta bancaria.

Para casos como el nombrado recientemente, se utiliza HTTPS el cual es una versión segura de HTTP, ya que utiliza cifrado basado en la seguridad de textos para crear un canal seguro.

De este modo se consigue que la información valiosa no pueda ser interceptada por ninguna persona ajena a la “conversación”.

- 5. Describa cómo opera un cliente HTTP (por ejemplo un navegador web) para recuperar una página HTML que contiene varios objetos. Analice la captura del archivo [captura_ejemplo_http.pcap](#) provisto por los docentes y represente el intercambio de**

mensajes mediante un gráfico ideado por Ud. ¿Qué primitivas se utilizan en cada caso?.

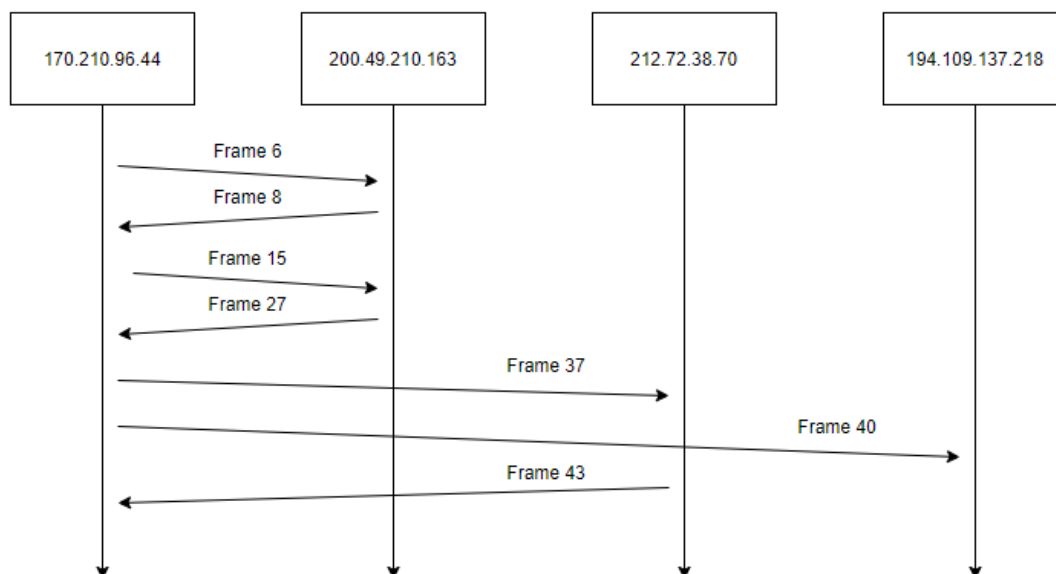
Para comenzar con el análisis de la captura titulada “captura_ejemplo_http.pcap” utilicé el comando:

```
tshark -r captura_ejemplo_http.pcap
```

Una vez visualizadas las tramas podemos concluir lo siguiente:

1. En la trama 1 se hace uso del protocolo DNS para que la máquina A (cuya IP es 170.210.96.44) haga una consulta a su resolver (cuya IP es 170.210.96.1) para obtener la dirección IP del servidor web al cual le va a realizar peticiones, en este caso www.unlu.edu.ar.
2. En la trama 2, el resolver le devuelve la dirección IP del servidor web deseado a la máquina A, la cual es 200.49.210.163.
3. Luego, en las tramas 3, 4 y 5, se establece una conexión TCP entre la máquina A y el servidor web nombrado anteriormente (Se puede visualizar el Three-way Handshake → SYN, SYN/ACK, ACK).
4. Una vez establecida la conexión, la máquina A procede a realizar una petición HTTP, la cual se observa en la trama 6 como: GET /prueba-tyr.htm HTTP/1.1.
5. En la respuesta recibida (en la trama 8), el navegador interpreta el lenguaje HTML (text/html) y visualiza que, entre las etiquetas, puede haber referencias hacia otros recursos que están alojados en otros dominios, por lo que el browser recurre nuevamente al protocolo DNS para traducir dichos dominios y luego realizar las peticiones necesarias para mostrar la página en el formato correcto.
6. Dichas peticiones son visibles en la trama 15 (GET /logo4c.gif HTTP/1.1), en la trama 37 (GET /i/logo3.gif HTTP/1.1) y en la trama 40 (GET /Pics/debian.jpg HTTP/1.1).

En todas las peticiones se utiliza la misma primitiva → GET.



6. ¿Qué es un servidor Proxy? ¿En qué situaciones se implementa? Brinde ejemplos.

Un servidor proxy es un servidor que recibe peticiones del User Agent (del cliente) y las realizan en lugar de tener que realizarla el cliente.

Una ventaja de la utilización de proxys es que se puede limitar los sitios en los cuales el usuario puede o no solicitar recursos.

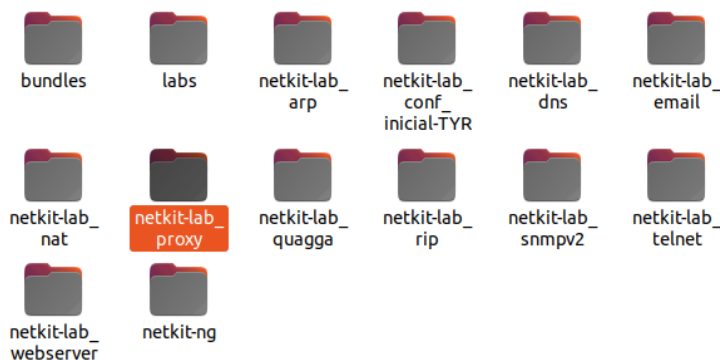
Otra ventaja de usar proxy es que cachea recursos, por lo que cuando un cliente realiza una petición, almacena la respuesta para el caso en que otro usuario realice la misma petición. Cuando ocurre la petición, el servidor proxy no tiene que "ir a buscar" la respuesta al servidor que la posee si no que la tiene almacenada.

Una posible implementación podría ser si tengo un servicio de internet cuya tasa de transferencia es limitada, ya que me beneficiaría mucho el uso de proxys, sacando provecho de los recursos cacheados.

Otra posible situación en la que podría ser beneficioso el uso de proxys puede ser una red de trabajo en la cual podría limitar el uso de páginas maliciosas por parte de los empleados o incluso limitar el acceso a redes sociales las cuales podrían llenar de improductividad.

7. Instale e inicie en el entorno Netkit el laboratorio de proxy HTTP provisto por los docentes, disponible en http://www.unlu.edu.ar/~tyr/netkit/netkit-lab_proxy-TYR.tar.gz

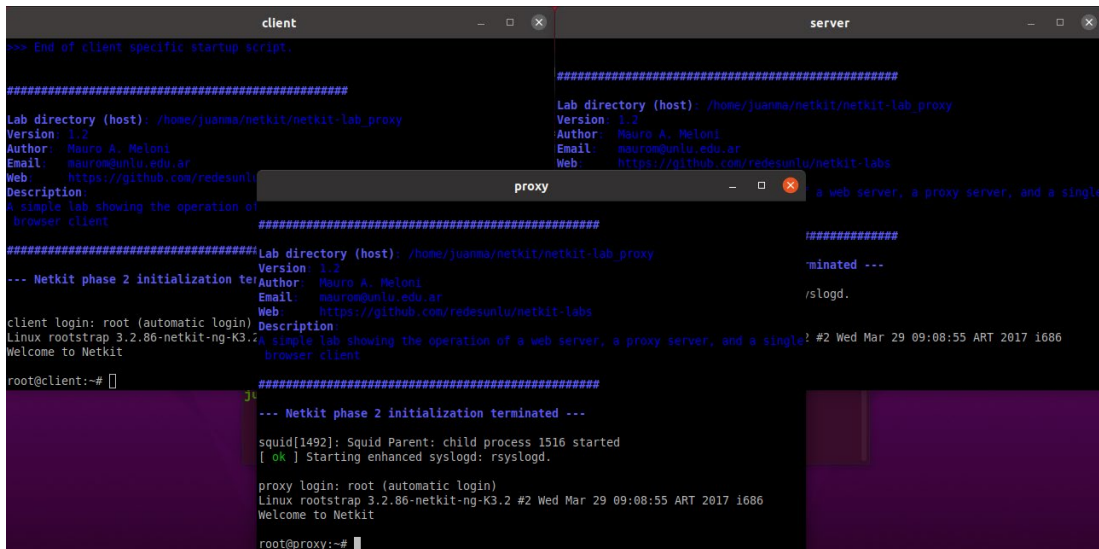
Para comenzar con la realización del ejercicio, primero ingresé al link provisto en el enunciado y descargué el laboratorio, situándolo posteriormente en la carpeta contenedora de todos los labs (/home/juanma/netkit).



Una vez hecho esto, me posiciono con la terminal dentro de la carpeta contenedora e inicializo el laboratorio con el comando

```
lstart
```

Obteniendo lo siguiente:

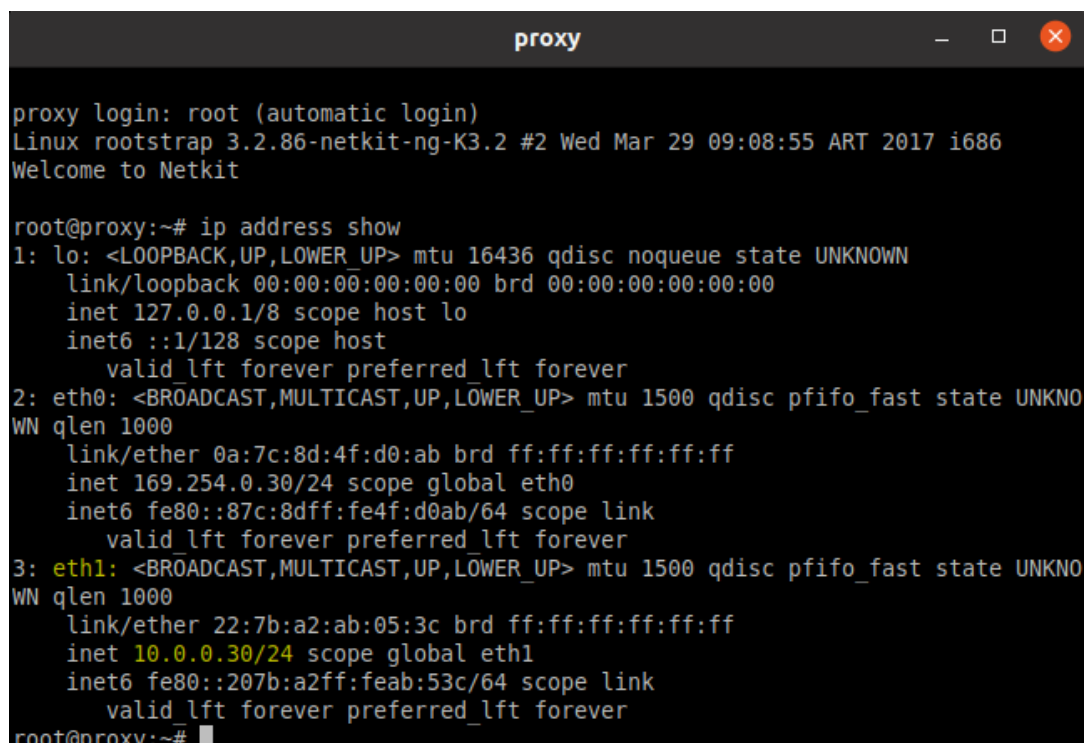


Este laboratorio comprende tres hosts: uno actúa como servidor web (con el servicio Apache2 en ejecución), uno actúa como cliente web (con el navegador Lynx instalado) y uno actúa como proxy HTTP (con el servicio Squid en ejecución).

- a. Averigüe y tome nota de la dirección IP asignada al Proxy en su interfaz **eth1**.

Para realizar esto, basta con utilizar el comando:

```
ip address show (o ip a s)
```



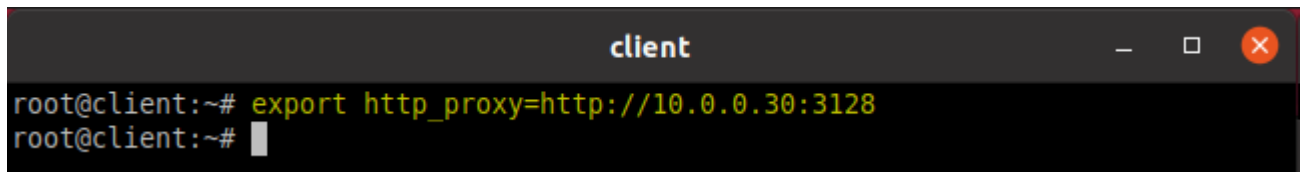
Como se aprecia en la imagen, la ip asignada al proxy en su interfaz eth1 es 10.0.0.30

- b. En el cliente, configure que las peticiones HTTP se realicen a través de proxy. Para ello, establezca la variable de entorno **HTTP_PROXY** como se muestra a continuación. Esto es equivalente a configurar las variables “Servidor proxy” y “Puerto” en los navegadores gráficos y en los celulares.

```
export http_proxy=http://DIRECCION_IP_PROXY:3128
```

Bien, sabiendo que la dirección IP del proxy es 10.0.0.30, para configurar la variable de entorno HTTP_PROXY basta con ejecutar dentro del cliente el siguiente comando:

```
export http_proxy=http://10.0.0.30:3128
```



```
client
root@client:~# export http_proxy=http://10.0.0.30:3128
root@client:~#
```

- c. Inicie una captura desde el host redirigiéndola a un archivo para su posterior análisis:

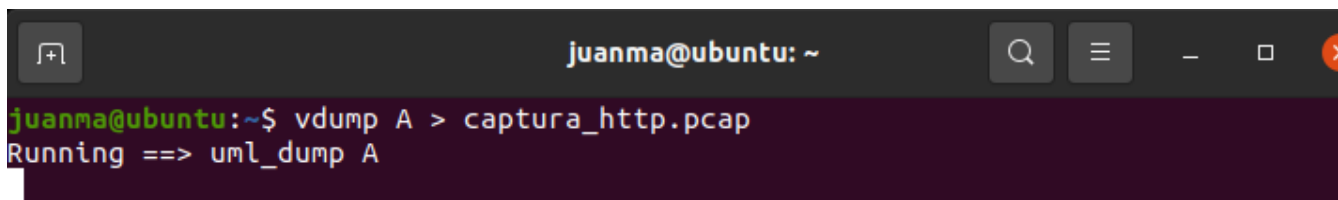
vdump A > nombre_archivo.pcap

Bien, como se indica en el enunciado, para iniciar una captura desde el host basta con posicionarme en su terminal y utilizar el comando

```
vdump A > nombre_archivo.pcap
```

En mi caso, utilizaré el siguiente comando:

```
vdump A > captura_http.pcap
```



```
juanma@ubuntu: ~
juanma@ubuntu:~$ vdump A > captura_http.pcap
Running ==> uml_dump A
```

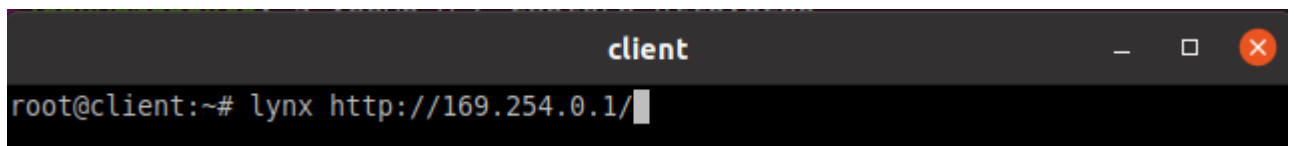
- d. En el cliente, navegue hacia la dirección **http://169.254.0.1/** utilizando un browser de consola:

lynx http://169.254.0.1/

Dicha dirección IP es la correspondiente al host servidor web.

Para realizar dicho procedimiento, voy a utilizar el browser de consola que trae instalado el laboratorio (lynx), con el siguiente comando:

```
lynx http://169.254.0.1/
```

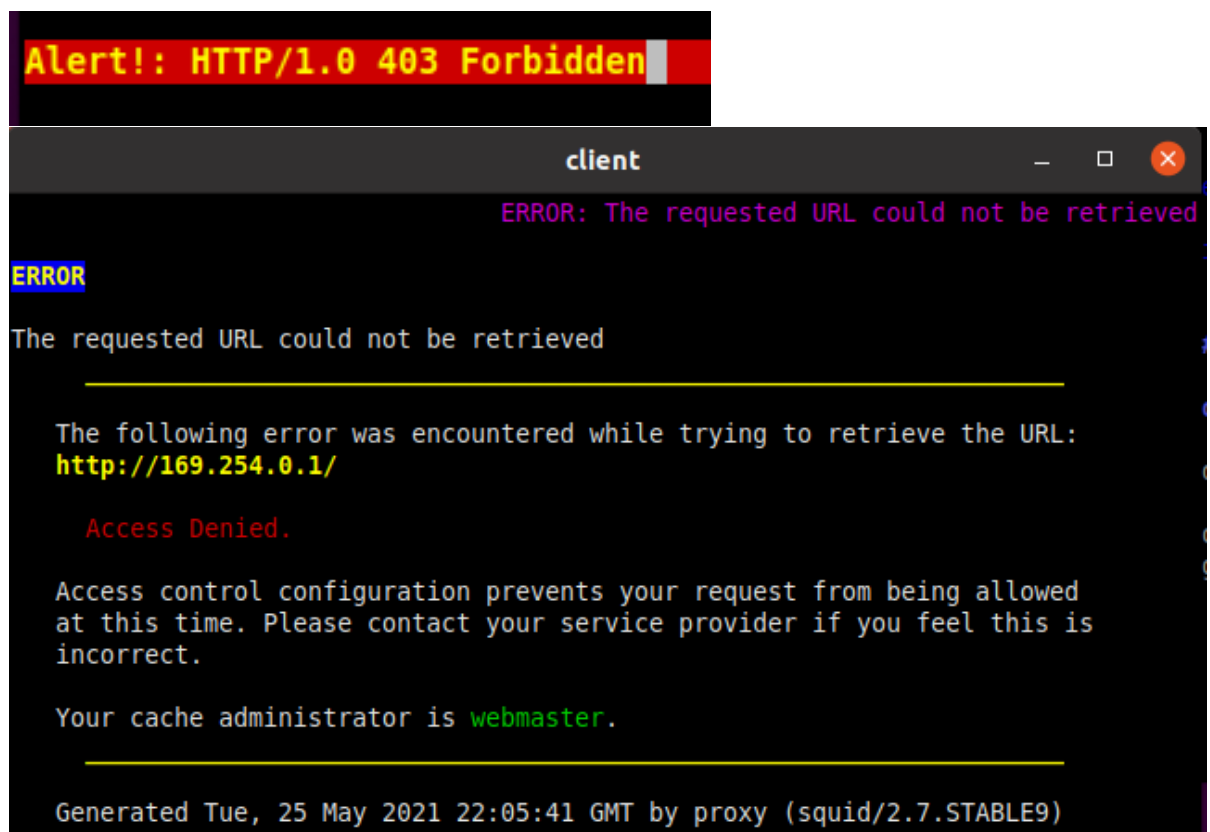


```
client
root@client:~# lynx http://169.254.0.1/
```

- e. Detenga la captura y analice el mensaje que aparece en pantalla. ¿Qué código de respuesta HTTP se retornó? Cierre el navegador web pulsando la tecla q

Para detener la captura, basta con ir a la terminal donde la inicié y presionar CTRL + C.

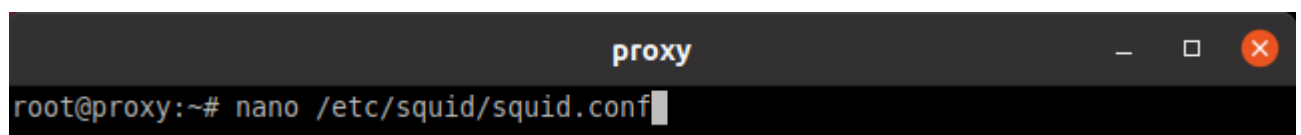
El código de respuesta que se retornó fue el código 403 Forbidden, lo que indica que el servidor ha entendido nuestra petición, pero se niega a autorizarla.



- f. La configuración de fábrica del software proxy Squid impide que los clientes naveguen a través de él. Para resolverlo, busque el archivo **/etc/squid/squid.conf** dentro del host proxy, edítelo y reemplace la línea **# http_access allow localnet** por **http_access allow localnet**

Para realizar esto, utilicé el editor de texto nano, con el siguiente comando:

```
nano /etc/squid/squid.conf
```



Una vez encontrada la línea que dice **# http_access allow localnet** procedo a modificarla.

```
# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
#http_access allow localnet
http_access allow localhost
```

Dejándola así:

```
# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
http_access allow localnet
http_access allow localhost
```

- g. Guarde los cambios y reinicie el proceso que actúa como proxy mediante el comando

service squid restart

De esta manera, el software Squid admitirá peticiones que realicen clientes que estén accediendo desde redes privadas (10.0.0.0/8, 192.168.0.0/16 y otras).

Una vez modificado todo, guardo los cambios en nano con CTRL + O.

Luego, reinicio el proceso con el comando:

```
service squid restart
```

```
root@proxy:~# service squid restart
[ ok ] Restarting Squid HTTP proxy: squid.
root@proxy:~#
```

- h. Inicie una nueva captura y vuelva a realizar la petición del punto d.

Para iniciar una nueva captura, vuelvo a utilizar el comando:

```
vdump A > nombre_captura.pcap
```

En este caso, utilizo el comando:

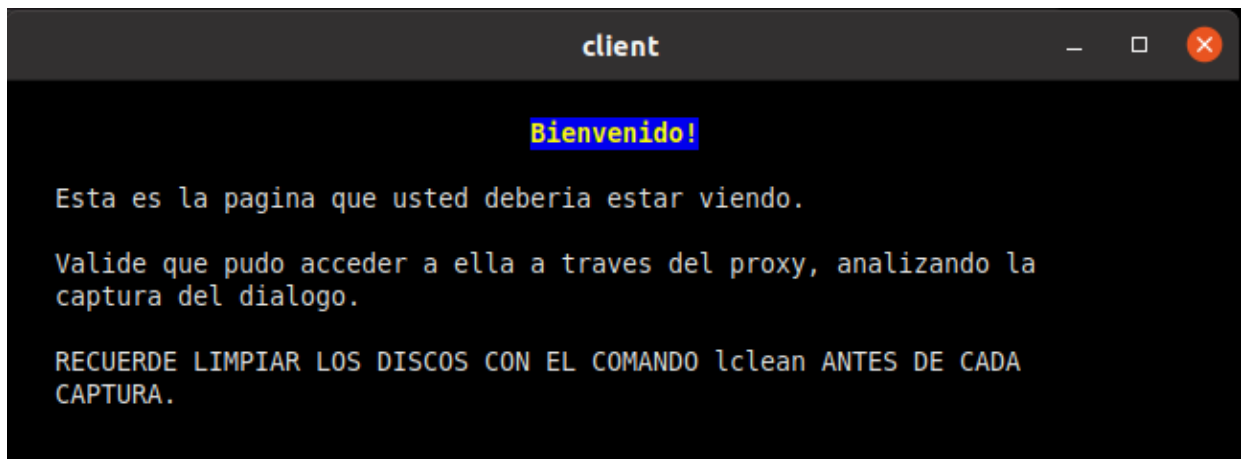
```
vdump A > captura_http_nueva.pcap
```

```
juanma@ubuntu:~$ vdump A > captura_http_nueva.pcap
Running ==> uml_dump A
```

Luego, realizo la petición del punto d con el comando:

```
lynx http://169.254.0.1/
```

Obteniendo el siguiente resultado:



- i. Indique qué mensaje aparece en pantalla. Cierre el navegador web pulsando la tecla q.

El mensaje que aparece en pantalla es el siguiente:

Bienvenido!

Esta es la página que usted debería estar viendo.

Valide que pudo acceder a ella a través del proxy, analizando la captura del dialogo.

RECUERDE LIMPIAR LOS DISCOS CON EL COMANDO lclean ANTES DE CADA CAPTURA.

Detenga la captura, analícela y responda:

1. ¿Qué encabezados envía el cliente al proxy-http en la petición?

Para analizar que encabezados envía el cliente al proxy-http utilizo el comando:

```
tshark -r captura_http_nueva.pcap -O http
```

En la trama número 4 se pueden visualizar los encabezados enviados por parte del cliente:

```
GET http://169.254.0.1/ HTTP/1.0\r\n
[Expert Info (Chat/Sequence): GET http://169.254.0.1/
HTTP/1.0\r\n]
[GET http://169.254.0.1/ HTTP/1.0\r\n]
[Severity level: Chat]
[Group: Sequence]
Request Method: GET
Request URI: http://169.254.0.1/
Request Version: HTTP/1.0
Host: 169.254.0.1\r\n
Accept: text/html, text/plain, text/css, text/sgml,
*/*;q=0.01\r\n
Accept-Encoding: gzip, compress, bzip2\r\n
```

```
Accept-Language: en\r\n
User-Agent: Lynx/2.8.8dev.12 libwww-FM/2.14 SSL-MM/1.4.1
GNUTLS/2.12.20\r\n
\r\n
[Full request URI: http://169.254.0.1/]
[HTTP request 1/1]
```

2. ¿Qué encabezados envía el proxy-http al servidor web en la petición?

Para visualizar los comandos enviados desde el proxy-http al servidor web utilizo el comando:

```
tshark -r captura_http_nueva.pcap -O http
```

En la trama 7 se pueden visualizar los encabezados:

```
HTTP/1.0 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.0 200 OK\r\n]
[HTTP/1.0 200 OK\r\n]
[Severity level: Chat]
[Group: Sequence]
Response Version: HTTP/1.0
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK

Date: Tue, 25 May 2021 22:25:00 GMT\r\n
Server: Apache/2.2.22 (Debian)\r\n
Last-Modified: Wed, 12 Apr 2017 14:58:57 GMT\r\n
ETag: "8f94-116-54cf9729cce40"\r\n
Accept-Ranges: bytes\r\n
Vary: Accept-Encoding\r\n
Content-Encoding: gzip\r\n
Content-Length: 224\r\n
[Content length: 224]
Content-Type: text/html\r\n
Age: 137\r\n
X-Cache: HIT from proxy\r\n
X-Cache-Lookup: HIT from proxy:3128\r\n
Via: 1.1 proxy:3128 (squid/2.7.STABLE9)\r\n
Connection: close\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.002391000 seconds]
[Request in frame: 4]
[Request URI: http://169.254.0.1/]
Content-encoded entity body (gzip): 224 bytes -> 278 bytes
File Data: 278 bytes
```

3. Mencione las diferencias que observa en los encabezados respecto a no utilizar un proxy-http (punto 4)

Al utilizar un proxy se puede visualizar que se añaden algunos campos que no estaban presentes en ejercicios anteriores como por ejemplo Via, Cache-Control y X-Forwarded-For.

El primero de ellos (Via) informa el servidor proxy que realizó la petición.

El segundo de ellos (Cache-Control) especifica las reglas que se van a seguir, como por ejemplo la cantidad máxima de tiempo por la que se almacena un recurso antes de consultar nuevamente al servidor.

Por último, el tercer campo (X-Forwarded-For) es utilizado por los proxys para indicar pasos intermedios a lo largo de la cadena de solicitud o respuesta, en donde se indica su URL.

4. ¿Es posible cambiar el número de puerto TCP en el que escucha el servidor proxy? ¿Qué línea del archivo de configuración hay que cambiar para que Squid escuche por conexiones en el puerto 8080?

Si, es posible cambiar el número de puerto TCP en el que escucha el servidor proxy.

Para realizar esto, además de modificar el puerto de proxy debo volver a configurar todos los hosts que accedían a este proxy con el número de puerto “mas reciente” en el cual se realizarán las peticiones.

Para que Squid escuche por conexiones en el puerto 8080 debo modificar la siguiente línea:

[Dentro del apartado NETWORK OPTIONS del archivo squid.conf]

```
#      If you run Squid on a dual-homed machine with an internal
#      and an external interface we recommend you to specify the
#      internal address:port in http_port. This way Squid will only be
#      visible on the internal address.
#
# Squid normally listens to port 3128
http_port 3128
```

La línea a modificar es la línea que se encuentra marcada.

8. ¿Cómo un sistema que realiza caché local puede determinar si algún objeto en el servidor original fue modificado con respecto a la copia actual, sin realizar la transferencia completa del objeto?

Como se pudo apreciar durante la realización del ejercicio anterior, los sistemas que poseen una caché local imponen para la regulación de la misma un tiempo de expiración, el cual una vez cumplido, los recursos que estén almacenados dejan de tener validez para futuras peticiones.

Cuando el sistema recibe una solicitud del lado del cliente, lo primero que se hace es intentar localizar el recurso en su caché local. Si se encuentra, se contrasta la fecha y hora de la versión de la página solicitada con el servidor.

Si la página no se ha modificado desde que se cargó, se devuelve al cliente inmediatamente, ahorrando mucho tráfico dado que solo se envía un paquete por la red (para comprobar la

versión). Para lograr esto se puede utilizar el campo contenido en la cabecera denominado "if-Modified-Since" utilizado en conjunto con el método GET. Este campo contiene un parámetro de fecha-hora y el recurso se va a transferir solo si se ha modificado desde la fecha-hora especificada, lo que permite que se realicen actualizaciones de manera eficaz de la caché, emitiendo de manera periódica peticiones GET a un servidor origen y recibiendo solo respuestas "pequeñas" a menos que haga falta una actualización.

9. ¿Qué es la interfaz CGI? ¿Para qué se utiliza?

Antes de definir la interfaz CGI veo necesario definir el concepto de documento dinámico, que no es más que un documento que no existe en un formato predefinido, es decir, se crean en el servidor web por medio de la ejecución de un programa cuando el navegador solicita el documento.

Al ser creados en cada petición, el contenido de los mismos puede variar.

La interfaz CGI (Common Gateway Interface) es una tecnología que se encarga de crear y manejar documentos dinámicos.

Es básicamente un acuerdo entre el webserver y una serie de procesos que define como va a ser el intercambio de información para que el webserver le entregue a ese proceso información de una petición HTTP y ese proceso le retorna al webserver que es lo que tiene que devolverle al User Agent.

Resumiendo, permite que un webserver y un proceso escrito en cualquier lenguaje de programación dialoguen dentro de la misma plataforma. El webserver recibe una petición, se la pasa a ese proceso, el proceso realiza cierto procesamiento, retorna una respuesta y el webserver se la envía al navegador.

10. ¿De qué formas un programa puede recibir parámetros por medio de la interfaz CGI? Comente las diferencias en el modo de operación en cada caso.

En primer lugar de definen una serie de variables que tienen como finalidad permitir el envío de datos a la interfaz CGI.

Se pueden clasificar en 2 grupos:

Variables específicas del servidor:

- GATEWAY_INTERFACE → Versión CGI.
- SERVER_NAME → Nombre del equipo del servidor.
- SERVER_SOFTWARE → Nombre y versión del servidor web.

Variables específicas de la petición:

- SERVER_PORT → Puerto TCP.
- SERVER_PROTOCOL → Versión HTTP.
- REQUEST_METHOD → Nombre del método HTTP.
- PATH_INFO → Sufijo de la ruta.
- SCRIPT_NAME → Ruta relativa al programa.
- REMOTE_HOST → Nombre del host del cliente.
- REMOTE_ADDR → Dirección IP del cliente.
- QUERY_STRING → Se usa para enviar datos del formulario web usando el método GET.

- CONTENT_LENGTH → Tamaño de los datos de entrada.
- CONTENT_TYPE → Tipo de contenido de la aplicación.

Se puede referenciar a un CGI usando formularios que contienen variables de entorno que sirven para el intercambio de información:

- Por medio del uso de la primitiva GET, obtenemos la variable de entorno QUERY_STRING en la cual se incluyen los parámetros dentro de la URL.
Por ejemplo:

GET http://server/cgi-bin/fichero.cgi?parámetro=valor

Este procedimiento se puede realizar en Spring (Framework de Java) mediante el uso de la anotación @RequestParam.

Luego del '?' se especifica la dupla parámetro=valor. En caso de ser mas de uno, se separan por '&'.

- Por medio del uso de la primitiva POST, se incluyen las variables dentro del cuerpo del mensaje y se reciben en la entrada estándar del CGI.
Se hace uso de las variables de entorno llamadas CONTENT_TYPE y CONTENT_LENGTH.

Para los ejercicios 11 a 14 deberá utilizar el intérprete de lenguaje Python versión 3.x disponible para múltiples plataformas y sistemas operativos. En los sistemas operativos Linux el intérprete usualmente está instalado por defecto. Si bien no es estrictamente necesario ninguna introducción “fuerte” en el lenguaje, se recomienda leer los primeros capítulos del tutorial en español indicado en las referencias de este trabajo.

11. Ejecute el siguiente comando en la consola de Linux, en el home del usuario (/home/alumno):

```
alumno@lab1:~$ python -m http.server
```

Primero, ejecuto el comando:

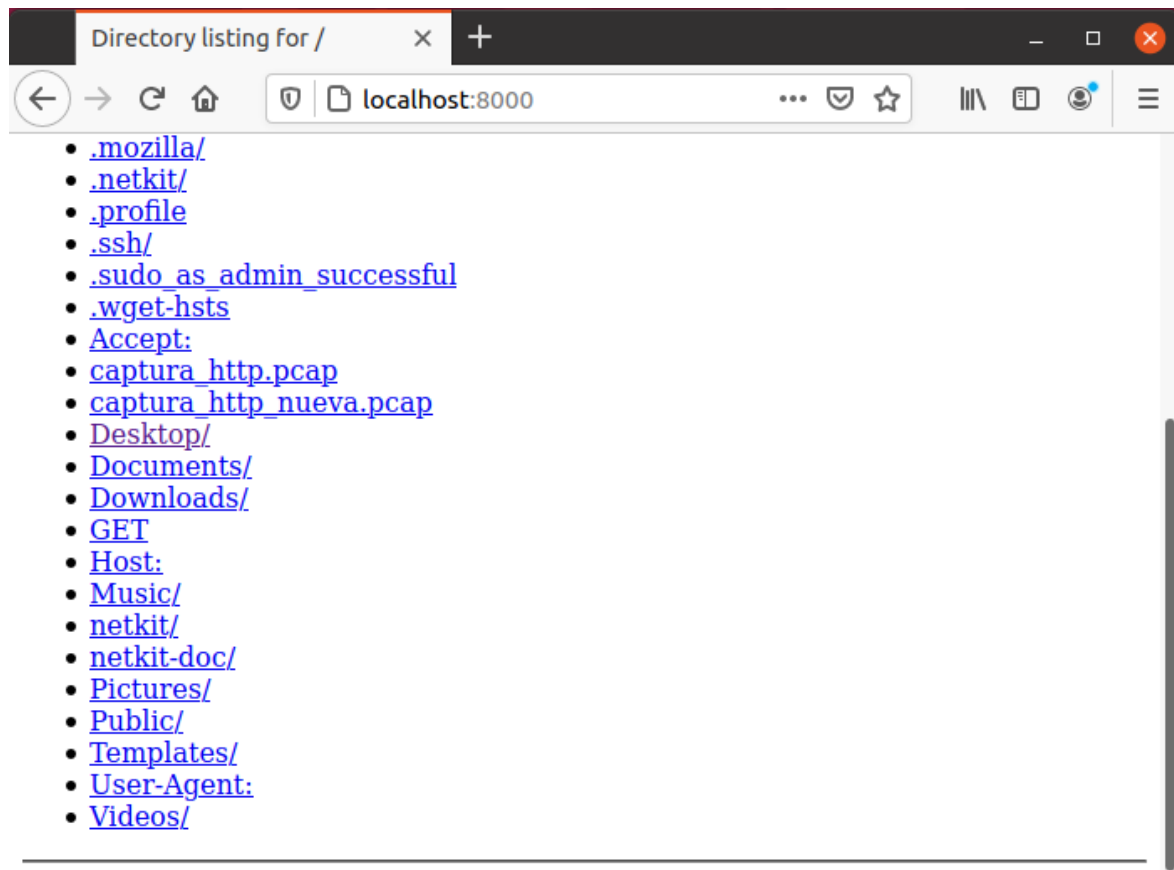
```
python3 -m http.server
```

```
juanma@ubuntu:~$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Luego, abra un navegador web e ingrese a la URL <http://localhost:8000>

a. ¿Qué es lo que se ve en el navegador?

Al ingresar a la URL <http://localhost:8000>, en el navegador se puede visualizar nuestro sistema de archivos.



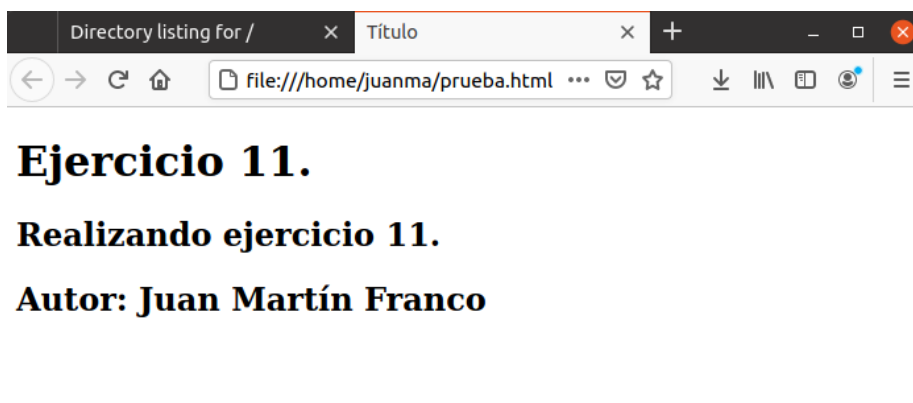
- b. ¿Cuál es la salida por consola del programa Python? ¿Qué puede interpretar de ella?

```
juanma@ubuntu: ~  
juanma@ubuntu:~$ python3 -m http.server  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...  
127.0.0.1 - - [25/May/2021 19:51:33] "GET / HTTP/1.1" 200 -
```

Se puede interpretar que hay un servicio corriendo en el puerto 8000.

Luego, se realiza una petición la cual resulta exitosa (la devolución del directorio), ya que se puede visualizar el código de estado 200.

- c. Abra un archivo de extensión **.html** en el directorio home del usuario. Si no existiera alguno, genere uno escrito por usted. ¿Qué sucede al abrirlo con el navegador web?



Al abrirlo con el navegador web se interpreta correctamente.

12. Escriba en un editor de texto el script en página siguiente y guárdelo en el archivo **http1.py**

Es importante destacar al momento de escribir el código, que en el lenguaje Python los espacios son utilizados para definir el nivel de anidamiento de la sentencia (ya que como se ve, no se utilizan llaves); por lo tanto, debe respetarse la sintaxis y todos los espacios del ejemplo.

Ejecute el script servidor HTTP con el comando **python http1.py** Luego, abra un navegador web e ingrese a la URL **http://localhost:8000**

```
#!/usr/bin/env python
# coding: utf-8

import http.server

HOST_NAME = 'localhost'
PORT = 8000

class Handler(http.server.BaseHTTPRequestHandler):

    def do_GET(server):
        """Respondo a una petición de tipo GET"""
        # Imprimo los encabezados
        print('-' * 80)
        print(server.command, server.path, server.request_version)
        print(server.headers)
        # Devuelvo la respuesta
        server.send_response(200)
        server.end_headers()

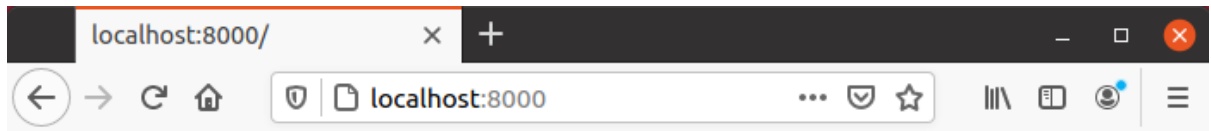
    def log_message(*args):
        """ Deshabilito la salida por defecto del servidor """
        pass

if __name__ == '__main__':
    http_server = http.server.HTTPServer((HOST_NAME, PORT), Handler)
    print('Ejecutando Server HTTP - http://%s:%s' % (HOST_NAME, PORT))
    print('Presione Ctrl+C para detener...')
    try:
        http_server.serve_forever()
    except KeyboardInterrupt:
        pass
    http_server.server_close()
    print('Deteniendo Server HTTP - %s:%s' % (HOST_NAME, PORT))
```

a. ¿Qué es lo que se ve en el navegador?

Dentro del navegador no se visualiza nada.

Solo se visualiza una pestaña en blanco como la siguiente:



- b. ¿Cuál es la salida por consola del servidor http? ¿Qué puede interpretar de la misma?

La salida por consola es la siguiente:

```
juanma@ubuntu: ~/Desktop
juanma@ubuntu:~/Desktop$ python3 http1.py
Ejecutando Server HTTP - http://localhost:8000
Presione Ctrl + C para detener...
-----
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Se puede interpretar que se solicita el recurso a localhost:8000, indicando el navegador, los tipos de respuesta aceptables, su codificación y se solicita que la conexión se mantenga activa.

- c. Experimente y compare las diferencias en la salida del servidor web:
- Con otros navegadores o clientes http.

En este caso, opté por acceder a la misma URL pero desde Chromium.

La diferencia que puedo apreciar es que, en la segunda petición, se puede visualizar el encabezado referer, el cual permite al cliente especificar la dirección del recurso desde el cual se obtuvo la URI de la solicitud actual.

```
-----
GET /favicon.ico HTTP/1.1
Host: localhost:8000
Connection: keep-alive
sec-ch-ua: "Chromium";v="93", " Not;A Brand";v="99"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4523.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:8000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

- Recargando la página con Ctrl+F5.

```
-----  
GET /favicon.ico HTTP/1.1  
Host: localhost:8000  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0  
Accept: image/webp,*/*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: keep-alive  
Referer: http://localhost:8000/  
Pragma: no-cache  
Cache-Control: no-cache
```

En este caso, aparecen los campos: Pragma: no-cache y Cache-Control: no-cache.

El primero de ellos se usa para incluir directivas específicas de implementación a lo largo de la cadena de petición/respuesta.

El segundo campo tiene la misma semántica que el primero, ambos impiden la posibilidad de que se utilice información cacheada.

- Ingresando a la URL <http://localhost:8000/test/de/url/>

```
-----  
GET /favicon.ico HTTP/1.1  
Host: localhost:8000  
Connection: keep-alive  
sec-ch-ua: "Chromium";v="93", " Not;A Brand";v="99"  
sec-ch-ua-mobile: ?0  
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4523.0 Safari/537.36  
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: no-cors  
Sec-Fetch-Dest: image  
Referer: http://localhost:8000/test/de/url/  
Accept-Encoding: gzip, deflate, br  
Accept-Language: en-US,en;q=0.9
```

En esta ocasión se envía una única petición.

En el navegador (Firefox o Chromium/Chrome), presione la combinación de teclas **Ctrl+Shift+I**, lo que da lugar a la apertura del depurador del navegador. Entre otras cosas, éste posee una pestaña o apartado llamado “Red”, que permite examinar cómo la página web es obtenida por el navegador, recurso por recurso. Seleccione haciendo clic sobre la petición hecha al servidor web local y describa qué encabezados devolvió el servidor escrito en Python.

▼ General

Request URL:

http://localhost:8000/

Request Method:

GET

Status Code:

● 200 OK

Remote Address:

127.0.0.1:8000

Referrer Policy:

strict-origin-when-cross-origin

▼ Response Headers

View source

Date:

Wed, 26 May 2021 03:57:32 GMT

Server:

BaseHTTP/0.6 Python/3.8.5

En este fragmento se puede apreciar que la respuesta a la petición realizada fue exitosa (código de estado 200 OK), la fecha y hora, el servidor y el protocolo utilizado.

13. Escriba en un editor de texto el siguiente script 1 y guárdelo en el archivo **http2.py**

```
#!/usr/bin/env python
# coding: utf-8
import http.server
HOST_NAME = 'localhost'
PORT = 8000

class Handler(http.server.BaseHTTPRequestHandler):
    def do_GET(server):
        """Respondo a una petición de tipo GET"""
        # Imprimo los encabezados
        print('-' * 80)
        print(server.command, server.path, server.request_version)
        print(server.headers)
        # Devuelvo la respuesta
        server.send_response(200)
        server.send_header('Content-Type', 'text/html')
        server.end_headers()
        server.wfile.write(b'<html><head><title>Pagina HTML de ejemplo</title>'
                           b'</head><body><p>Esta es una prueba, con texto en <b>negrita</b>,'
                           b'<i>cursiva</i> e incluso una imagen externa:</p>'
                           b'</body></html>')

    def log_message(*args):
        """Deshabilito la salida por defecto del servidor """
        pass

if __name__ == '__main__':
    http_server = http.server.HTTPServer((HOST_NAME, PORT), Handler)
    print('Ejecutando Server HTTP - http://%s:%s' % (HOST_NAME, PORT))
    print('Presione Ctrl+C para detener...')

    try:
        http_server.serve_forever()
    except KeyboardInterrupt:
        pass
    http_server.server_close()
    print('Deteniendo Server HTTP - %s:%s' % (HOST_NAME, PORT))
```

Ejecute el script servidor HTTP con el comando **python http2.py** Luego, abra un navegador web e ingrese a la URL **http://localhost:8000** con el depurador del navegador activo.

- a. Analice cómo se obtiene la página principal y la imagen embebida en el documento.

Al ingresar el comando `python3 http2.py` por consola y luego abrir en el navegador la dirección **http://localhost:8000** se puede visualizar lo siguiente:



La página principal se obtiene a través de la URL <http://localhost:8000/> encontrando código HTML que se interpreta por el navegador. Dentro de dicho código aparece una referencia a la ubicación de la imagen, la cual se obtiene desde <http://www.labredes.unlu.edu.ar/themes/glossyblue/images/>.

- b. **Detenga el servidor, modifique el encabezado Content-Type a text/plain y vuelva a ejecutar la prueba. ¿Qué apariencia tiene la página web y por qué?**

Una vez que fue modificado el encabezado a text/plain y fue realizada nuevamente la prueba, la página web muestra el texto del script, ya que, como le indicamos, el navegador va a interpretar esto como texto plano y no como código HTML.

14. Escriba en un editor de texto el siguiente script y guárdelo en el archivo **http3.py**

```
#!/usr/bin/env python
# coding: utf-8

import http.server

HOST_NAME = 'localhost'
PORT = 8000

def detectar_so(user_agent):
    # Ver listados en http://www.useragentstring.com/pages/useragentstring.php
    if 'Linux' in user_agent:
        return b'Veo que Ud. está usando Linux como S.O.'
    elif 'Windows' in user_agent:
        return b'Veo que Ud. está usando Windows como S.O.'
    else:
        return b'No conozco su S.O.'

def get_pagina_ok():
    """ Función que devuelve la página de éxito de ejemplo """
    # Como alternativa podría abrirse un archivo del disco, leerlo y devolverlo
    # como cadena de texto.
    return (b'<html><head><title>Pagina HTML de ejemplo</title>'
            b'</head><body><p>Esta es una prueba, con texto en <b>negrita</b>,'
            b'<i>cursiva</i> e incluso una imagen externa:</p>'
            b'')

class Handler(http.server.BaseHTTPRequestHandler):
    def do_GET(server):
```



```

"""Respondo a una petición de tipo GET"""
# Imprimo los encabezados por consola
print('-' * 80)
print(server.command, server.path, server.request_version)
print(server.headers)
# Devuelvo la respuesta
if server.path.startswith('/ir_a/'):
    ir_a = server.path.split('/')[1]
    server.send_response(302)
    server.send_header('Location', 'http://' + ir_a)
    server.end_headers()
elif server.path.startswith('/no_existe'):
    server.send_response(404)
    server.send_header('Content-Type', 'text/plain')
    server.end_headers()
    server.wfile.write('Pagina no encontrada')
else:
    server.send_response(200)
    server.send_header('Content-Type', 'text/html')
    server.end_headers()
    server.wfile.write(get_pagina_ok())
    server.wfile.write(bytes(b'<p>' + detectar_so(server.headers['User-Agent'])))
    server.wfile.write(b'</p></body></html>')

def log_message(*args):
    """Deshabilito la salida por defecto del servidor"""
    Pass

if __name__ == '__main__':
    http_server = http.server.HTTPServer((HOST_NAME, PORT), Handler)
    print('Ejecutando Server HTTP - http://%s:%s' % (HOST_NAME, PORT))
    print('Presione Ctrl+C para detener...')
    try:
        http_server.serve_forever()
    except KeyboardInterrupt:
        pass
    http_server.server_close()
    print('Deteniendo Server HTTP - %s:%s' % (HOST_NAME, PORT))

```

Ejecute el script servidor HTTP con el comando python **http3.py**. Luego, abra un navegador web e ingrese a la URL <http://localhost:8000> con el depurador del navegador activo.

Al ejecutar el script y abrir el navegador en dicha URL, se obtiene el siguiente resultado:

Esta es una prueba, con texto en **negrita**, *cursiva* e incluso una imagen externa:



Veo que Ud. esta usando Linux como S.O.

- a. ¿La página es siempre igual (estática) o dinámica? ¿De qué manera se intenta averiguar el Sistema Operativo del cliente?

La página es dinámica, ya que varía en función del sistema operativa de la máquina desde la cual estoy accediendo.

El sistema operativo se averigua analizando el campo User Agent en el método de la solicitud → Si se encuentra la palabra Linux (como pasó en mi caso) se indicará en la salida que estoy utilizando dicho sistema operativo.

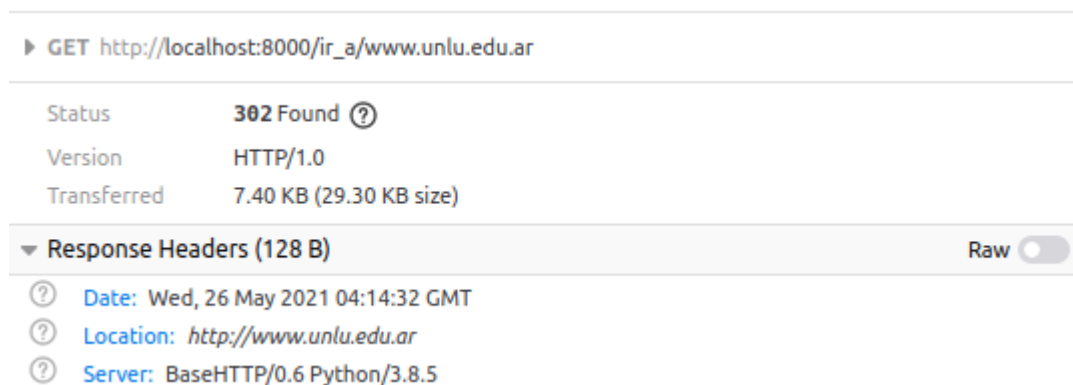
- b. ¿Qué sucede si se ingresa a la URL `http://localhost:8000/ir_a/www.unlu.edu.ar`? ¿Por qué? Analice el comportamiento con el depurador del navegador.

En este caso, al ingresar a dicha URL, me redirige a la página web de la UNLu.

Esto ocurre porque el script está configurado para que, si el path del servidor comienza con “/ir_a”, automáticamente se redirija a la dirección correspondiente, en este caso `www.unlu.edu.ar`.

En caso de no ser válido el dominio, se retornará el error correspondiente.

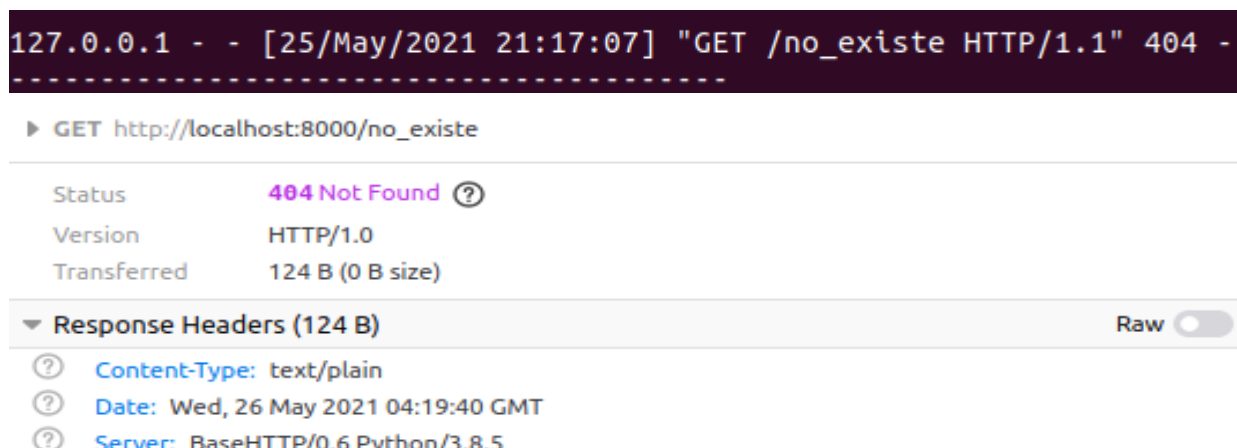
En el depurador del navegador obtenemos lo siguiente:



Se puede visualizar que se realiza la petición de tipo GET a la URL definida y que nos redirige a la página que está contenida en el campo “Location”.

El código de estado es 302 el cual indica que el recurso solicitado ha sido movido temporalmente a la URL dada por las cabeceras Location

- c. ¿Qué sucede si se ingresa a la URL `http://localhost:8000/no_existe`? ¿Por qué? Analice el comportamiento con el depurador del navegador.



En este caso, se puede visualizar un código de estado 404 el cual indica que la página no fue encontrada.