

Práctica 2

Introducción

2. ¿Cuál es la función de la capa de aplicación?

- Define el formato de los mensajes: Existen protocolos que trabajan de forma binaria, por ejemplo usando ASN y otros en forma textual ASCII como HTTP.
- Define la semántica de cada uno de los mensajes.
- Define como debe ser el dialogo (intercambio de mensajes). Que mensajes se deben intercambiar.
- Ejemplo concreto: Protocolo HTTP y sus implementaciones mediante servidores WEB y browsers (navegadores).

3. Si dos procesos deben comunicarse:

a. ¿Cómo podrían hacerlo si están en diferentes máquinas?

Dos procesos de sistemas terminales distintos se comunican mediante el intercambio de mensajes a través de una red de computadoras. Uno de los procesos actúa como emisor, creando y enviando mensajes a la red, mientras que el otro proceso actúa como receptor, recibiendo estos mensajes y, posiblemente, respondiendo con mensajes propios.

b. Y si están en la misma máquina, ¿qué alternativas existen?

Cuando los procesos se ejecutan en el mismo sistema terminal, pueden comunicarse entre sí mediante sistemas de comunicación interprocesos, aplicando reglas gobernadas por el sistema operativo del sistema terminal.

4. Explique brevemente cómo es el modelo Cliente/Servidor. De un ejemplo de un sistema Cliente/Servidor en la “vida cotidiana” y un ejemplo de un sistema informático que siga el modelo Cliente/Servidor. ¿Conoce algún otro modelo de comunicación?

Existe un host siempre activo, denominado servidor, con una dirección fija y conocida llamada dirección IP. Este servidor da (valga la redundancia) servicio a las solicitudes de muchos otros hosts, que son los clientes. Para ello el servidor recibe y sirve las solicitudes de los clientes (estos mismos no se comunican directamente entre sí).

Debido a que el servidor tiene una dirección fija y conocida, y siempre está activo, un cliente siempre puede contactar con él enviando un paquete a su dirección IP.

A menudo, en una aplicación cliente-servidor un único host servidor es incapaz de responder a todas las solicitudes de sus clientes. Por esta razón, en las arquitecturas cliente-servidor suele utilizarse un centro de datos, que alberga un gran número de hosts, para crear un servidor virtual de gran capacidad.

Un ejemplo de la vida cotidiana puede ser el intercambio entre cliente-vendedor en una tienda. Un ejemplo de un sistema informático que siga este modelo es una aplicación web, en donde el cliente sería el navegador web y el servidor sería el servidor web

Otro modelo de comunicación es P2P (Peer-to-Peer) en donde los dispositivos participantes son iguales y pueden actuar tanto como clientes como servidores para compartir recursos directamente entre sí, sin una infraestructura centralizada. Un ejemplo de P2P es BitTorrent.

5. Describa la funcionalidad de la entidad genérica “Agente de usuario” o “User agent”

Un User Agent es la aplicación o software que actúa en nombre del usuario en un sistema o red. Es un componente que representa al usuario en sus interacciones con servicios en línea, sitios web u otros sistemas.

En el contexto web, se refiere principalmente al navegador que un usuario emplea para acceder a sitios web. El User Agent comunica al servidor detalles del navegador, su versión, y a veces información sobre el sistema operativo y dispositivo. Esto permite que los sitios adapten su contenido a las capacidades del navegador.

En una solicitud HTTP del navegador, el User Agent es un encabezado que informa sobre el navegador y sistema operativo.

HTTP

6. Observe el índice de la RFC2616, busque el apartado donde se describe el requerimiento y la respuesta. ¿Qué son y en qué se diferencian HTML y HTTP? ¿En qué entidad ubicaría a HTML?

HTTP es el Protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol), que es el protocolo de la capa de aplicación de la Web. HTTP se implementa mediante dos programas: un programa cliente y un programa servidor. Ambos programas, que se ejecutan en sistemas terminales diferentes, se comunican entre sí intercambiando mensajes HTTP. HTTP define la

estructura de estos mensajes y cómo el cliente y el servidor intercambian los mensajes.

HTML, por otro lado, es el lenguaje de marcado utilizado para crear la estructura y el contenido de las páginas web. La mayoría de las páginas web están constituidas por un archivo base HTML y varios objetos referenciados. Por ejemplo, si una página web contiene texto HTML y cinco imágenes JPEG, entonces la página web contiene seis objetos: el archivo base HTML y las cinco imágenes. El archivo base HTML hace referencia a los otros objetos contenidos en la página mediante los URL de los objetos. Cada URL tiene dos componentes: el nombre de host del servidor que alberga al objeto y el nombre de la ruta al objeto.

El navegador web (cliente) es el que interpreta y renderiza el código HTML para mostrar la página web al usuario y el servidor web es el encargado de enviar los archivos HTML. El contenido HTML se encuentra en la entidad de respuesta, y es lo que el servidor envía al cliente como parte de la respuesta a la solicitud HTTP.

El envío y recibimiento de este mismo se hace a través de mensajes respetando el protocolo HTTP.

7. Utilizando la VM, abra una terminal e investigue sobre el comando curl. Analice para qué sirven los siguientes parámetros (-I, -H, -X, -s).

curl es una herramienta para transferir datos desde o hacia un servidor, utilizando uno de los protocolos admitidos (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET y TFTP). El comando está diseñado para trabajar sin interacción del usuario.

-I, --head

(HTTP FTP FILE) Obtiene solo los encabezados. Los servidores HTTP cuentan con el comando HEAD que utiliza para obtener nada más que el encabezado de un documento. Cuando se usa en un archivo FTP o FILE, curl muestra solo el tamaño del archivo y la hora de la última modificación.

-H, --header <header/@file>

Sirve para incluir un header adicional en la solicitud HTTP. Se puede especificar cualquier número de encabezados adicionales. Hay que tener en cuenta que si se agrega un encabezado personalizado que tiene el mismo nombre que uno de los internos que usaría curl, se usará el encabezado establecido externamente en lugar del interno.

-X, --request <command>

Especifica un método de solicitud personalizado que se utilizará al comunicarse con el servidor HTTP. CURL por defecto utiliza GET.

-s, --silent

Modo silencioso de CURL. No mostrará la barra de progreso ni los mensajes de error. Silencia Curl. Seguirá generando los datos que solicita,

potencialmente incluso al terminal/salida estándar, a menos que los redirija. Es útil si deseas que la salida de curl sea más limpia y solo quieras ver la respuesta del servidor

8. Ejecute el comando curl sin ningún parámetro adicional y acceda a www.redes.unlp.edu.ar. Luego responda:

a. ¿Cuántos requerimientos realizó y qué recibió? Pruebe redirigiendo la salida(>) del comando curl a un archivo con extensión html y abrirlo con un navegador

Realicé un requerimiento (método GET) y recibí un HTML.

b. ¿Cómo funcionan los atributos href de los tags link e img en html?

El atributo href en la etiqueta <link> se utiliza para enlazar recursos externos y el atributo src en la etiqueta se utiliza para especificar la ubicación de la imagen que se mostrará en la página web.

En ambos casos, el navegador realiza solicitudes HTTP/HTTPS para descargar los recursos externos especificados en las URLs de los atributos href o src. Luego, dependiendo del tipo de recurso, realiza acciones adicionales como aplicar estilos en el caso de las hojas de estilo o mostrar imágenes en el caso de las etiquetas .

c. Para visualizar la página completa con imágenes como en un navegador, ¿alcanza con realizar un único requerimiento? ¿Cuántos requerimientos serían necesarios para obtener una página que tiene dos CSS, dos Javascript y tres imágenes? Diferencie como funcionaría un navegador respecto al comando curl ejecutado previamente.

No, no alcanza con un único requerimiento. Cada recurso, como un archivo CSS, un archivo JavaScript o una imagen, se solicita por separado al servidor.

Se necesitarán:

- 1 requerimiento para el archivo base HTML
- 1 requerimiento para el favicon
- 2 requerimientos para CSS
- 2 requerimientos para Javascript
- 3 requerimientos para las tres imágenes

Si el HTML tiene algún atributo href también se realizarán los requerimientos correspondientes para cada atributo.

curl solo realizara un solo requerimiento (el archivo base HTML), requiere solicitudes manuales.

9. Ejecute a continuación los siguientes comandos:

```
curl -v -s www.redes.unlp.edu.ar > /dev/null
curl -I -v -s www.redes.unlp.edu.ar
```

-v muestra información del intercambio que se está haciendo, información detallada de la comunicación entre el cliente y el servidor. (> es lo que el cliente le envió al servidor, < lo que el servidor respondió)

- Observe la salida y luego repita la prueba, pero previamente inicie una nueva captura en wireshark. Utilice la opción Follow Stream. ¿Qué se transmitió en cada caso?
- ¿A que se debió esta diferencia entre lo que se transmitió y lo que se mostró en pantalla?

```
curl -v -s www.redes.unlp.edu.ar > /dev/null
```

GET / HTTP/1.1

Host: www.redes.unlp.edu.ar

User-Agent: curl/7.74.0

Accept: */*

HTTP/1.1 200 OK

Date: Mon, 28 Aug 2023 16:09:12 GMT

Server: Apache/2.4.56 (Unix)

Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT

ETag: "1322-5f7457bd64f80"

Accept-Ranges: bytes

Content-Length: 4898

Content-Type: text/html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>...Redes y Comunicaciones...Facultad de Informática...UNLP...</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="description" content="">

<meta name="author" content="">

<!-- Le styles -->

<link href="/bootstrap/css/bootstrap.css" rel="stylesheet">

<link href="/css/style.css" rel="stylesheet">

<link href="/bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

<!-- HTML5 shim, for IE6-8 support of HTML5 elements -->

```
<!--[if lt IE 9]>
<script src="./bootstrap/js/html5shiv.js"></script>
<![endif]-->
</head>
```

```
<body>
<div id="wrap">
<div class="navbar navbar-inverse navbar-fixed-top">
<div class="navbar-inner">
<div class="container">
<a class="brand" href="./index.html"><i class="icon-home icon-white"></i></a>
<a class="brand" href="https://catedras.info.unlp.edu.ar" target="_blank">Redes y
Comunicaciones</a>
<a class="brand" href="http://www.info.unlp.edu.ar" target="_blank">Facultad de
Informática</a>
<a class="brand" href="http://www.unlp.edu.ar" target="_blank">UNLP</a>
</div>
</div>
</div>
```

```
<div class="container">
```

```
<!-- Main hero unit for a primary marketing message or call to action -->
<div class="hero-unit">
<h2>Bienvenidos a Redes y Comunicaciones!</h2>
<p>Este CD es parte de los enunciados prácticos de la materia Redes y
Comunicaciones de la carrera de Licenciatura en Informática de la UNLP para
la cursada del presente año y servir como herramienta para la
realización de los trabajos prácticos.</p>
</div>
```

```
<div class="row">
<div class="span12">
<h3>Acerca de la VM</h3>
<p>Esta máquina virtual está basada en Debian GNU/Linux y fue
creada por la cátedra de Redes y Comunicaciones de la carrera de Licenciatura
en Informática de la UNLP para incluir las herramientas y configuraciones que
se utilizarán a lo largo de la cursada.</p>
<p>Se ha configurado al usuario <em><strong>root</strong></em> y al usuario
<em><strong>redes</strong></em> con la misma contraseña:
<strong>redes</strong>.</p>
</div>
</div>
<div class="row">
<div class="span12">
<h3>Ejercicios Prácticos</h3>
<p>Todo el material se va a encontrar publicado en el sitio de la cátedra en <a
href="https://catedras.info.unlp.edu.ar/"
target="_blank">https://catedras.info.unlp.edu.ar/</a>.</p>
</div>
```

```

</div>
<div class="row">
<div class="span2">
<h4>Introducci&oslash;n</h4>
<p>
<ul>
<li>Nociones b&acutes;icas</li>
</ul>
</p>
</div>
<div class="span3">
<h4>Capa de Aplicaci&oslash;n</h4>
<p>
<ul>
<li><a href="http/protocolos.html">Protocolos HTTP</a></li>
<li><a href="http/metodos.html">M&eacute;todos HTTP</a></li>
</ul>
</p>
</div>
<div class="span3">
<h4>Capa de Transporte</h4>
<p>
<ul>
<li>TCP</li>
<li>UDP</li>
</ul>
</p>
</div>
<div class="span2">
<h4>Capa de Red</h4>
<p>
<ul>
<li>IP</li>
<li>
Algoritmos de ruteo:<br/>
Topolog&iacute;as CORE:
<ul>
<li><a href="/core/1-ruteo-estatico.xml" target="_blank">Est&acutes;tico</a></li>
<li><a href="/core/2-ruteo-RIP.xml" target="_blank">RIP</a></li>
<li><a href="/core/3-ruteo-OSPF.xml" target="_blank">OSPF</a></li>
</ul>
</li>
<li>ICMP</li>
</ul>
</p>
</div>
<div class="span2">
<h4>Capa de Enlace</h4>
<p>
<ul>

```

```
</li>ARP</li>
</li>
Switch - Hub
</li>
</ul>
</p>
</div>
</div>
```

```
<div class="row">
<hr>
<p>Desarrollado originalmente por Christian Rodriguez y Paula Venosa en el a.o 2007,
modificado en 2013 por el grupo de desarrollo de Lihuen GNU/Linux y en 2016 y 2020
por Leandro Di Tommaso.</p>
<br/>
</div>
</div>
</div>
<div id="footer">
<div class="container">
<p class="muted credit">Redes y Comunicaciones</p>
</div>
</div>
```

```
</body>
</html>
```

```
curl -I -v -s www.redes.unlp.edu.ar
```

```
HEAD / HTTP/1.1
Host: www.redes.unlp.edu.ar
User-Agent: curl/7.74.0
Accept: */*
```

```
HTTP/1.1 200 OK
Date: Mon, 28 Aug 2023 16:12:19 GMT
Server: Apache/2.4.56 (Unix)
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "1322-5f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 4898
Content-Type: text/html
```

La diferencia se debe a `> /dev/null` que redirige la salida estandar(la respuesta enviada por el servidor) a `/dev/null` (que la descarta)

10. Investigue cómo define las cabeceras la RFC.

a. Establece todas las cabeceras posibles?.

La RFC no establece todas las cabeceras posibles ya que se pueden utilizar cabeceras personalizadas, pero proporciona una lista de cabeceras estándar y comunes que se utilizan en el protocolo HTTP. Algunas de estas cabeceras incluyen "Date", "Content-Type", "Content-Length", "User-Agent", "Host", "Accept", "Authorization" y muchas más. Los desarrolladores tienen la flexibilidad de definir cabeceras personalizadas según las necesidades de sus aplicaciones, pero se espera que sigan las convenciones y estándares definidos en la RFC.

b. ¿Cuántas cabeceras viajaron en el requerimiento y en la respuesta del ejercicio anterior?

3 en el requerimiento
7 en la respuesta

c. ¿La cabecera Date es una de las definidas en la RFC? ¿Qué indica?

Sí, la cabecera "Date" es una de las cabeceras definidas en la RFC. Esta cabecera indica la fecha y hora en que se generó la solicitud o respuesta HTTP.

11. Utilizando curl, realice un requerimiento con el método HEAD al sitio www.redes.unlp.edu.ar e indique:

a. ¿Qué información brinda la primer línea de la respuesta?

La primera línea brinda la versión de HTTP (HTTP/1.1), el código de estado (200) y la reason phrase (OK)

b. ¿Cuántos encabezados muestra la respuesta?

7

c. ¿Qué servidor web está sirviendo la página?

Apache/2.4.56 (Unix)

d. ¿El acceso a la página solicitada fue exitoso o no?

Si, fue exitoso ya que el código de estado es 200 OK, que es el código de respuesta de estado satisfactorio

e. ¿Cuándo fue la última vez que se modificó la página?

El 19 de marzo de 2023 a las 16:04:46 (19:04:46 GMT)

- f. Solicite la página nuevamente con curl usando GET, pero esta vez indique que quiere obtenerla sólo si la misma fue modificada en una fecha posterior a la que efectivamente fue modificada. ¿Cómo lo hace? ¿Qué resultado obtuvo? ¿Puede explicar para qué sirve?

Lo hago haciendo uso del header "If-Modified-Since":<fecha>

```
curl -H "If-Modified-Since:Sun, 19 Mar 2023 19:04:46 GMT"
www.redes.unlp.edu.ar
```

```
redes@debian:~$ curl -I -H "If-Modified-Since:Sun, 19 Mar 2023 19:04:46 GMT" www
.redes.unlp.edu.ar
HTTP/1.1 304 Not Modified
Date: Tue, 29 Aug 2023 16:52:50 GMT
Server: Apache/2.4.56 (Unix)
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "1322-5f7457bd64f80"
Accept-Ranges: bytes
```

Si se quiere acceder a una página recién cuando sea modificada y esta pesara (mucho), nos ahorramos todo el intercambio en el caso de que no haya sido modificada, ya que es la misma que tendríamos guardada.

12. En HTTP/1.0, ¿cómo sabe el cliente que ya recibió todo el objeto solicitado completamente? ¿Y en HTTP/1.1?

En HTTP/1.0:

El cliente se da cuenta de que ha recibido todo el objeto solicitado cuando el servidor cierra la conexión después de enviar la respuesta.

En HTTP/1.1:

Se introducen mejoras para manejar este problema. El encabezado "Content-Length" indica al cliente la longitud en bytes del objeto en la respuesta, permitiéndole saber cuántos datos esperar. El encabezado "Transfer-Encoding" con valor "chunked" divide la respuesta en trozos, y el cliente detecta el final de la respuesta cuando recibe un trozo de tamaño 0.

13. Investigue los distintos tipos de códigos de retorno de un servidor web y su significado en la RFC. ¿Qué parte se ve principalmente interesada de esta información, cliente o servidor? ¿Es útil que esté detallado y clasificado en la RFC?. Dentro de la VM, ejecute los siguientes comandos y evalúe el estado que recibe

- curl -I <http://unlp.edu.ar>
- curl -I www.redes.unlp.edu.ar/restringido/index.php
- curl -I www.redes.unlp.edu.ar/noexiste

●	1XX	Códigos informativos	El servidor ha recibido la petición y procederá con ella.
●	2XX	Códigos de éxito	El servidor ha recibido, entendido y procesado la solicitud correctamente.
●	3XX	Códigos de redirección	El servidor ha recibido la solicitud, pero hay una redirección a alguna otra parte (o, en raras ocasiones, alguna acción adicional que debe completarse).
●	4XX	Códigos de error de cliente	El servidor no puede encontrar (o alcanzar) la página o la web. Se trata de un error del lado de la web.
●	5XX	Códigos de error de servidor	El cliente ha realizado una solicitud válida, pero el servidor ha fallado al completarla.

Ejemplos:

- 200 OK
- 301 Moved Permanently
- 400 Bad Request
- 403 Access Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 500 Internal Server Error (CGI Error)
- 501 Method Not Implemented

Tanto los clientes como los servidores se benefician al entender los códigos de respuesta en el proceso de comunicación web. El cliente puede saber si su solicitud fue procesada con éxito o si hubo algún error, mientras que el servidor utiliza el código de retorno para comunicar el resultado de la solicitud al cliente y de alguna manera orientarlo sobre cómo interpretar la respuesta.

Si, es útil que este detallado ya que establece un estándar común para la comunicación entre clientes y servidores en la web asegurando respuestas coherentes e interoperables de acuerdo a la situación.

```

redes@debian:~$ curl -I http://unlp.edu.ar
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0
Date: Tue, 29 Aug 2023 17:13:05 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://unlp.edu.ar/

redes@debian:~$ curl -I www.redes.unlp.edu.ar/restringido/index.php
HTTP/1.1 401 Unauthorized
Date: Tue, 29 Aug 2023 17:13:23 GMT
Server: Apache/2.4.56 (Unix)
WWW-Authenticate: Basic realm="Authentication Required"
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "cb-5f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 203
Content-Type: text/html

redes@debian:~$ curl -I www.redes.unlp.edu.ar/noexiste
HTTP/1.1 404 Not Found
Date: Tue, 29 Aug 2023 17:13:35 GMT
Server: Apache/2.4.56 (Unix)
Content-Type: text/html; charset=iso-8859-1

```

14. Utilizando curl, acceda al sitio

www.redes.unlp.edu.ar/restringido/index.php y siga las instrucciones y las pistas que vaya recibiendo hasta obtener la respuesta final. Será de utilidad para resolver este ejercicio poder analizar tanto el contenido de cada página como los encabezados

```

redes@debian:~$ curl http://www.redes.unlp.edu.ar/restringido/index.php
<h1>Acceso restringido</h1>

<p>Para acceder al contenido es necesario autenticarse. Para obtener los datos de acceso seguir las instrucciones detalladas en www.redes.unlp.edu.ar/obtener-usuario.php</p>
redes@debian:~$ curl www.redes.unlp.edu.ar/obtener-usuario.php
<p>Para obtener el usuario y la contraseña haga un requerimiento a esta página seteando el encabezado 'Usuario-Redes' con el valor 'obtener'</p>
redes@debian:~$ curl -H "Usuario-Redes:obtener" www.redes.unlp.edu.ar/obtener-usuario.php
<p>Bien hecho! Los datos para ingresar son:

    Usuario: redes

    Contraseña: RYC

    Ahora vuelva a acceder a la página inicial con los datos anteriores.

    PISTA: Investigue el uso del encabezado Authorization para el método Basic. El comando base64 puede ser de ayuda!</p>
redes@debian:~$ echo -n "redes:RYC" | base64
cmVkZXh6M6U1LD

redes@debian:~$ curl -v -H "Authorization: Basic cmVkZXh6M6U1LD" http://www.redes.unlp.edu.ar/restringido/index.php
* Trying 172.28.0.50:80...
* Connected to www.redes.unlp.edu.ar (172.28.0.50) port 80 (#0)
> GET /restringido/index.php HTTP/1.1
> Host: www.redes.unlp.edu.ar
> User-Agent: curl/7.74.0
> Accept: */*
> Authorization: Basic cmVkZXh6M6U1LD
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 302 Found
< Date: Tue, 29 Aug 2023 17:22:15 GMT
< Server: Apache/2.4.56 (Unix)
< X-Powered-By: PHP/7.4.33
< Location: http://www.redes.unlp.edu.ar/restringido/the-end.php
< Content-Length: 230
< Content-Type: text/html; charset=UTF-8
<
<h1>Excelente!</h1>

<p>Para terminar el ejercicio deberás agregar en la entrega los datos que se muestran en la siguiente página.</p>
<p>ACLARACIÓN: la URL de la siguiente página está contenida en esta misma respuesta.</p>
* Connection #0 to host www.redes.unlp.edu.ar left intact

```

```

redes@debian:~$ curl -v -H "Authorization: Basic cmVkZXN6M6UllD" http://www.redes.unlp.edu.ar/restringido/the-end.php
* Trying 172.28.0.50:80...
* Connected to www.redes.unlp.edu.ar (172.28.0.50) port 80 (#0)
> GET /restringido/the-end.php HTTP/1.1
> Host: www.redes.unlp.edu.ar
> User-Agent: curl/7.74.0
> Accept: */*
> Authorization: Basic cmVkZXN6M6UllD
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 29 Aug 2023 17:22:43 GMT
< Server: Apache/2.4.56 (Unix)
< X-Powered-By: PHP/7.4.33
< Content-Length: 159
< Content-Type: text/html; charset=UTF-8
<
¡Felicitaciones, llegaste al final del ejercicio!
Fecha: 2023-08-29 17:22:43
* Connection #0 to host www.redes.unlp.edu.ar left intact

```

***NOTAS:**

El método de autenticación "**Basic**" en HTTP es una forma simple de autenticar a los usuarios utilizando credenciales (nombre de usuario y contraseña). Se utiliza en combinación con HTTPS para mayor seguridad.

El motivo por el cual las credenciales se codifican en base64 es principalmente para evitar problemas con caracteres especiales que podrían afectar la comunicación HTTP. Base64 es una forma de representar datos binarios en una cadena de caracteres ASCII, lo que lo hace seguro para transmitir a través de HTTP.

El encabezado "**Location**" en HTTP se utiliza para redirigir al cliente a una ubicación diferente. Cuando un servidor envía una respuesta con un encabezado "Location", está indicando al cliente que realice una nueva solicitud a la URL especificada en ese encabezado.

15. Utilizando la VM, realice las siguientes pruebas:

- a. Ejecute el comando 'curl www.redes.unlp.edu.ar/extras/prueba-http-1-0.txt' y copie la salida completa (incluyendo los dos saltos de línea del final).
- b. Desde la consola ejecute el comando telnet www.redes.unlp.edu.ar 80 y luego pegue el contenido que tiene almacenado en el portapapeles. ¿Qué ocurre luego de hacerlo?

```

redes@debian:~$ telnet www.redes.unlp.edu.ar 80
Trying 172.28.0.50...
Connected to www.redes.unlp.edu.ar.
Escape character is '^['.
GET /http/HTTP-1.1/ HTTP/1.0
User-Agent: curl/7.38.0
Host: www.redes.unlp.edu.ar
Accept: */*

HTTP/1.1 200 OK
Date: Tue, 29 Aug 2023 17:38:03 GMT
Server: Apache/2.4.56 (Unix)
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "7605f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 1888
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Protocolo HTTP: versiones</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="">
  <meta name="author" content="">

  <!-- Le styles -->
  <link href=".../bootstrap/css/bootstrap.css" rel="stylesheet">
  <link href=".../css/style.css" rel="stylesheet">
  <link href=".../bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

  <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
  <!--[if lt IE 9]>
    <script src=".../bootstrap/js/html5shiv.js"></script>
  <![endif]>
</head>

<body>

  <div id="wrap">

    <div class="navbar navbar-inverse navbar-fixed-top">
      <div class="navbar-inner">
        <div class="container">
          <a class="brand" href=".../index.html"><i class="icon-home icon-white"></i></a>
          <a class="brand" href="https://catedras.info.unlp.edu.ar" target="blank">Redes y Comunicaciones</a>
          <a class="brand" href="http://www.info.unlp.edu.ar" target="blank">Facultad de Informácutica</a>
          <a class="brand" href="http://www.unlp.edu.ar" target="blank">UNLP</a>
        </div>
      </div>
    </div>

    <div class="container">
      <h1>Ejemplo del protocolo HTTP 1.1</h1>
      <p>
        Esta páacutegina se visualiza utilizando HTTP 1.1. Utilizando el capturador de paquetes analice cuantos flujos utiliza el navegador para visualizar la páacutegina con sus imáacutegenes en contraposicióacut;n con el protocolo HTTP/1.0.
      </p>
      </div>
    </div>

    <div class="container">
      <h2>Imagen de ejemplo</h2>
      
    </div>
  </div>

```

- c. Repita el proceso anterior, pero copiando la salida del recurso /extras/prueba-http-1-1.txt. Verifique que debería poder pegar varias veces el mismo contenido sin tener que ejecutar telnet nuevamente.

El resultado es el mismo, la diferencia que hay es que la conexión no se cierra una vez que el servidor me envía la respuesta.

16. En base a lo obtenido en el ejercicio anterior, responda:

- ¿Qué está haciendo al ejecutar el comando telnet?

El comando intenta establecer una conexión a través del protocolo Telnet con el servidor en "www.redes.unlp.edu.ar" en el puerto 80.

- ¿Qué lo diferencia con curl? Observe la definición de método y recurso en la RFC. Luego responda, ¿Qué método HTTP utilizó?

Telnet se centra en la comunicación interactiva con servidores a través del protocolo Telnet, curl se centra en la transferencia de datos utilizando diversos protocolos, incluidos los seguros.

Se utilizó el método GET

GET /http/HTTP-1.1/ HTTP/1.0

User-Agent: curl/7.38.0

Host: www.redes.unlp.edu.ar

Accept: */*

- ¿Qué recurso solicitó? ¿Qué diferencias notó entre los dos casos? ¿Puede explicar por qué?

Solicito el recurso /http/HTTP-1.1/. La diferencia que hay es que la conexión no se cierra una vez que el servidor envía la respuesta en el punto c. Esto se debe a que en contraste con HTTP 1.0, donde cada solicitud y respuesta requería abrir y cerrar una nueva conexión, en HTTP 1.1, una sola conexión puede ser reutilizada para varias transacciones.

- **¿Cuál de los dos casos le parece más eficiente? Piense en el ejercicio donde analizó la cantidad de requerimientos necesarios para obtener una página con estilos, javascripts e imágenes. El caso elegido, ¿puede traer asociado algún problema?**

Es más eficiente el caso de HTTP 1.1 ya que reduce la sobrecarga asociada con la apertura y el cierre de conexiones, lo que conduce a una carga más rápida de páginas web y una mejor utilización de los recursos de red.

Hay un problema asociado con la conexión persistente, especialmente cuando se trata de recursos grandes o de larga duración. Si un recurso se mantiene en la conexión durante un período prolongado, podría bloquear la conexión y retrasar la obtención de otros recursos. Esto se conoce como el problema de "holgura de cabeza" (head-of-line blocking). Para abordar este problema, los navegadores modernos utilizan técnicas como la multiplexación y el pipelining para permitir la transferencia de múltiples recursos a la vez.

17. En el siguiente ejercicio veremos la diferencia entre los métodos POST y GET. Para ello, será necesario utilizar la VM y la herramienta Wireshark. Antes de iniciar considere:

- **Capture los paquetes utilizando la interfaz con IP 172.28.0.1. (Menú "Capture ->Options". Luego seleccione la interfaz correspondiente y presione Start).**
- **Para que el analizador de red sólo nos muestre los mensajes del protocolo http introduciremos la cadena 'http' (sin las comillas) en la ventana de especificación de filtros de visualización (display-filter). Si no hiciéramos esto veríamos todo el tráfico que es capaz de capturar nuestra placa de red. De los paquetes que son capturados, aquel que esté seleccionado será mostrado en forma detallada en la sección que está justo debajo. Como sólo estamos interesados en http ocultaremos toda la información que no es relevante para esta práctica (Información de trama, Ethernet, IP y TCP). Desplegar la información correspondiente al protocolo HTTP bajo la leyenda "Hypertext Transfer Protocol".**
- **Para borrar la cache del navegador, deberá ir al menu "Herramientas->Borrar historial reciente". Alternativamente puede utilizar Ctrl+F5 en el navegador para forzar la petición HTTP evitando el uso de caché del navegador.**

- En caso de querer ver de forma simplificada el contenido de una comunicación http, utilice el botón derecho sobre un paquete HTTP perteneciente al flujo capturado y seleccione la opción Follow TCP Stream.
- a. Abra un navegador e ingrese a la URL: www.redes.unlp.edu.ar e ingrese al link en la sección “Capa de Aplicación” llamado “Métodos HTTP”. En la página mostrada se visualizan dos nuevos links llamados: Método GET y Método POST. Ambos muestran un formulario como el siguiente:

Nombre

Apellido

Email

Sexo Masculino: ☒ Femenino: ☐

Contraseña

Recibir confirmaciones por email ☐

Enviar

- b. Analice el código HTML.

GET

```
<!DOCTYPE html>
<html lang="en">
  <head> </head>
  <body>
    <div id="wrap">
      <div class="navbar navbar-inverse navbar-fixed-top"> </div>
      <div class="container">
        <div>
          <h1>Métodos HTTP: GET</h1>
        </div>
        <form method="GET" action="metodos-lectura-valores.php"> </form>
      </div>
    </div>
  </body>
</html>
```

POST

```
<!DOCTYPE html>
<html lang="en">
  <head> </head>
  <body>
    <div id="wrap">
      <div class="navbar navbar-inverse navbar-fixed-top"> </div>
      <div class="container">
        <div>
          <h1>Métodos HTTP: POST</h1>
        </div>
        <form method="POST" action="metodos-lectura-valores.php"> </form>
      </div>
    </div>
  </body>
</html>
```

- c. Utilizando el analizador de paquetes Wireshark capture los paquetes enviados y recibidos al presionar el botón Enviar

GET

GET /http/metodos-lectura-
valores.php?form_nombre=Agustina&form_apellido=Rojas&form_mail=mim
ail&form_sexo=sexo_fem&form_pass=123&form_confirma_mail=on
HTTP/1.1
Host: www.redes.unlp.edu.ar
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101
Firefox/91.0
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
Referer: http://www.redes.unlp.edu.ar/http/metodo-get.html
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Wed, 30 Aug 2023 14:22:02 GMT
Server: Apache/2.4.56 (Unix)
X-Powered-By: PHP/7.4.33
Content-Length: 2620
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>M&eacute;todos HTTP: Lectura de valores desde
REQUEST</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">

    <!-- Le styles -->
    <link href="../bootstrap/css/bootstrap.css" rel="stylesheet">
    <link href="../css/style.css" rel="stylesheet">
    <link href="../bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

    <!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
      <script src="../bootstrap/js/html5shiv.js"></script>
    <![endif]-->
  </head>

  <body>

    <div id="wrap">

      <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="navbar-inner">
```

```

<div class="container">
  <a class="brand" href="../index.html"><i class="icon-home icon-
white"></i></a>
  <a class="brand" href="https://catedras.info.unlp.edu.ar"
target="_blank">Redes y Comunicaciones</a>
  <a class="brand" href="http://www.info.unlp.edu.ar"
target="_blank">Facultad de Inform&acute;tica</a>
  <a class="brand" href="http://www.unlp.edu.ar"
target="_blank">UNLP</a>
</div>
</div>
</div>

```

```

<div class="container">

```

```

<h1>M&eacute;todos HTTP: Lectura de valores desde REQUEST</h1>

```

```

<p>

```

```

  &Eacute;sta p&acute;gina es el resultado de leer los valores recibidos
  en un mensaje HTTP. Desde el punto de vista del programador, puede
  obligarse la lectura seg&uacute;n el m&eacute;todo sea <em>GET, POST u
  ambos</em>. En este caso, utilizamos el &uacute;ltimo caso, es decir que
  este mismo c&acute;digo sirve para leer valores enviados utilizando GET
  o POST indistintamente.

```

```

  <h2> Los valores recibidos son </h2>

```

```

  <table border="0" width="80">

```

```

    <tr>

```

```

      <td nowrap>Nombre:</td><td>Agustina </td>

```

```

    </tr>

```

```

    <tr>

```

```

      <td nowrap>Apellido:</td><td>Rojas </td>

```

```

    </tr>

```

```

    <tr>

```

```

      <td nowrap>Email:</td><td>mimail </td>

```

```

    </tr>

```

```

    <tr>

```

```

      <td nowrap>Sexo:</td><td>Femenino </td>

```

```

    </tr>

```

```

    <tr>

```

```

      <td nowrap>Contrase&ntilde;a:</td><td>123 </td>

```

```

    </tr>

```

```

    <tr>

```

```

      <td nowrap>Recibir confirmaciones por email:</td><td>Si </td>

```

```

    </tr>

```

```

  </table>

```

```

</p>

```

```

</div>

```

```

</div>

```

```

<div id="footer">

```

```

  <div class="container">

```

```
<p class="muted credit">Redes y Comunicaciones</p>
</div>
</div>
</body>
</html>
```

POST

```
POST /http/metodos-lectura-valores.php HTTP/1.1
Host: www.redes.unlp.edu.ar
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101
Firefox/91.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 112
Origin: http://www.redes.unlp.edu.ar
Connection: keep-alive
Referer: http://www.redes.unlp.edu.ar/http/metodo-post.html
Upgrade-Insecure-Requests: 1
```

```
form_nombre=Agustina&form_apellido=Rojas&form_mail=mimail&form_sex
o=sexo_fem&form_pass=123&form_confirma_mail=on
```

```
HTTP/1.1 200 OK
Date: Wed, 30 Aug 2023 14:26:02 GMT
Server: Apache/2.4.56 (Unix)
X-Powered-By: PHP/7.4.33
Content-Length: 2620
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>M&eacute;todos HTTP: Lectura de valores desde
REQUEST</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="">
<meta name="author" content="">

<!-- Le styles -->
<link href="../bootstrap/css/bootstrap.css" rel="stylesheet">
<link href="../css/style.css" rel="stylesheet">
<link href="../bootstrap/css/bootstrap-responsive.css" rel="stylesheet">

<!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
<!--[if lt IE 9]>
```

```

<script src="./bootstrap/js/html5shiv.js"></script>
<![endif]-->
</head>

<body>

<div id="wrap">

<div class="navbar navbar-inverse navbar-fixed-top">
<div class="navbar-inner">
<div class="container">
<a class="brand" href=" ../index.html"><i class="icon-home icon-
white"></i></a>
<a class="brand" href="https://catedras.info.unlp.edu.ar"
target="_blank">Redes y Comunicaciones</a>
<a class="brand" href="http://www.info.unlp.edu.ar"
target="_blank">Facultad de Inform&acute;tica</a>
<a class="brand" href="http://www.unlp.edu.ar"
target="_blank">UNLP</a>
</div>
</div>
</div>

<div class="container">

<h1>M&eacute;todos HTTP: Lectura de valores desde REQUEST</h1>
<p>
&Eacute;sta página es el resultado de leer los valores recibidos
en un mensaje HTTP. Desde el punto de vista del programador, puede
obligarse la lectura según el método sea <em>GET, POST u
ambos</em>. En este caso, utilizamos el último caso, es decir que
este mismo código sirve para leer valores enviados utilizando GET
o POST indistintamente.
<h2> Los valores recibidos son </h2>
<table border="0" width="80">
<tr>
<td nowrap>Nombre:</td><td>Agustina </td>
</tr>
<tr>
<td nowrap>Apellido:</td><td>Rojas </td>
</tr>
<tr>
<td nowrap>Email:</td><td>mimail </td>
</tr>
<tr>
<td nowrap>Sexo:</td><td>Femenino </td>
</tr>
<tr>
<td nowrap>Contraseña:</td><td>123 </td>
</tr>
<tr>

```

```

        <td nowrap>Recibir confirmaciones por email:</td><td>Si </td>
    </tr>
</table>
</p>
</div>

</div>

<div id="footer">
    <div class="container">
        <p class="muted credit">Redes y Comunicaciones</p>
    </div>
</div>
</body>
</html>

```

d. ¿Qué diferencias detectó en los mensajes enviados por el cliente?

La diferencia que se detecta es que si se trata de un GET, se envía la información como parámetro de la URL, mientras que si se trata de un POST, esta se envía en el cuerpo de la solicitud.

e. ¿Observó alguna diferencia en el browser si se utiliza un mensaje u otro?

Si, en el caso del GET, se pueden ver en la URL los parámetros enviados, en el caso del POST no, se mantiene la misma URL que se tenía antes de apretar el botón ENVIAR.

18. HTTP es un protocolo stateless, para sortear esta carencia muchos servicios se apoyan en el uso de Cookies. ¿En qué RFC se definió dicha funcionalidad? Investigue cuál es el principal uso que se le da a Set-Cookie y Cookie en HTTP. ¿Qué atributo de la RFC original fue en parte aprovechado para la implementación?

Las cookies, definidas en [RFC 6265], permiten a los sitios seguir la pista a los usuarios. Actualmente, la mayoría de los sitios web comerciales más importantes utilizan cookies.

La tecnología de las cookies utiliza cuatro componentes: (1) una línea de cabecera de la cookie en el mensaje de respuesta HTTP; (2) una línea de cabecera de la cookie en el mensaje de solicitud HTTP; (3) el archivo de cookies almacenado en el sistema terminal del usuario y gestionado por el navegador del usuario; y (4) una base de datos back-end en el sitio web.

El principal uso de las cabeceras "Set-Cookie" y "Cookie" en HTTP es permitir a los servidores y clientes mantener un estado persistente entre las solicitudes y respuestas, a pesar de la naturaleza stateless del protocolo HTTP.

La Cookie es introducida al cliente mediante el mensaje en el header **Set-Cookie**: mensaje que indica un par (nombre,valor).

El cliente en cada requerimiento luego de haber almacenado la Cookie se la enviara al servidor con el header **Cookie**:

La implementación de cookies se basa en gran parte en el atributo "Set-Cookie" de la RFC original (RFC 2109), que se centraba en establecer cookies y manejar su formato. Sin embargo, la RFC 6265 se introdujo para corregir algunas deficiencias y ambigüedades en el enfoque anterior y para mejorar la seguridad y la privacidad en el uso de cookies.

19. ¿Cuál es la diferencia entre un protocolo binario y uno basado en texto? ¿de que tipo de protocolo se trata HTTP/1.0, HTTP/1.1 y HTTP/2?

Un protocolo binario transmite datos en forma de patrones de bits que las máquinas pueden entender directamente. Suele ser más eficiente en términos de velocidad y uso de ancho de banda, ya que no se desperdicia espacio en caracteres de formato o en la interpretación de texto legible por humanos.

Un protocolo basado en texto envía datos como cadenas legibles utilizando caracteres como letras, números y símbolos. Estos caracteres suelen estar codificados en ASCII o UTF-8. Aunque esto hace que la comunicación entre sistemas sea más comprensible para los humanos, puede ser menos eficiente en términos de velocidad y uso de ancho de banda, ya que se requiere más información para representar los mismos datos que en formato binario.

HTTP/1.0, HTTP/1.1 son protocolos basados en textos mientras que HTTP/2 es un protocolo binario

20. Analice de que se tratan las siguientes características de HTTP/2: stream, frame, server-push

Un **stream** es un canal virtual bidireccional que opera dentro de una única conexión TCP. Cada solicitud y respuesta se intercambia a través de un stream separado. Cada stream posee un ID y una prioridad asignada. Los mensajes HTTP/2 (solicitudes y respuestas) se transmiten utilizando estos streams. Los mensajes HTTP/2 se dividen en **frames** que se envían a través del mismo stream. Básicamente, los streams son identificados y desglosados en frames, que constituyen la unidad de comunicación más pequeña. Estos frames se transmiten y se vuelven a ensamblar en el destino. Existen varios tipos de frames, como los de encabezado, de datos y de prioridad. Estos frames se combinan para formar las solicitudes y respuestas que se intercambian en los streams. Todos los streams coexisten en una misma conexión.

El **server-push** es una característica que permite al servidor enviar recursos adicionales al cliente antes de que este los solicite de manera explícita. Por ejemplo, pueden ser recursos como hojas de estilo CSS o scripts JavaScript. No siempre es lo que necesita el cliente, depende de que funcionalidad este ofrece.

21. Responder las siguientes preguntas:

- a. ¿Qué función cumple la cabecera Host en HTTP 1.1? ¿Existía en HTTP 1.0? ¿Qué sucede en HTTP/2? (Ayuda: <https://undertow.io/blog/2015/04/27/An-in-depth-overview-of-HTTP2.html> para HTTP/2)

La cabecera "Host" en HTTP 1.1 tiene la función de indicar el nombre de dominio (se puede indicar opcionalmente el puerto) al que se está haciendo la solicitud dentro del encabezado de la petición. La inclusión de la cabecera "Host" en una solicitud es obligatoria. Esta es sumamente importante en servidores que alojan múltiples sitios web en una misma dirección IP, ya que permite al servidor saber a qué sitio se está dirigiendo la solicitud. También es importante mencionar que las cachés proxy web necesitan la información proporcionada por la línea de cabecera del host.

En HTTP 1.0, la cabecera "Host" no es obligatoria pero puede ser necesaria dependiendo de cómo el servidor web esté configurado. Esto significa que si la cabecera "Host" no estaba presente en una solicitud HTTP 1.0, el servidor asumiría que la solicitud estaba destinada al dominio asociado con la dirección IP del servidor.

En HTTP/2, la cabecera "Host" es reemplazada por ":authority".

- b. ¿Cómo quedaría en HTTP/2 el siguiente pedido realizado en HTTP/1.1 si se está usando https?
GET /index.php HTTP/1.1
Host: www.info.unlp.edu.ar

Quedaría
:method: get
:path: /index.php
:scheme: https
:authority: www.info.unlp.edu.ar