Práctica 2

1) La capa de aplicación es donde residen las aplicaciones de red y sus protocolos. La capa de aplicación de Internet incluye muchos protocolos, tales como el protocolo HTTP (que permite la solicitud y transferencia de documentos web), SMTP (que permite la transferencia de mensajes de correo electrónico) y FTP (que permite la transferencia de archivos entre dos sistemas terminales). Provee servicios de comunicación a los usuarios y a las aplicaciones. Podría considerarse como una interfaz con el usuario u otros servicios. Brinda los protocolos que implementan las aplicaciones.

2) Los procesos de dos sistemas terminales diferentes se comunican entre ellos intercambiando mensajes a través de la red de computadoras. Un proceso emisor crea y envía mensajes a la red; un proceso receptor recibe estos mensajes y posiblemente responde devolviendo mensajes.
   Un proceso puede interpretarse como un programa que se ejecuta dentro de un sistema terminal. Cuando los procesos se ejecutan en el mismo sistemas terminal, pueden comunicarse entre sí mediante la comunicación entre procesos aplicando las reglas gobernadas por el sistema operativo del sistema terminal.

3) Una aplicación de red consta de parejas de procesos que se envían mensajes entre sí a través de una red. Por ejemplo, en la aplicación web, un proceso de un navegador cliente intercambia mensajes con un proceso de un servidor web. La carga es compartida, el cliente pone procesamiento de interfaz mientras que el servidor corre servicio esperando pasivamente la conexión. En el contexto de una sesión de comunicación entre una pareja de procesos, el proceso que inicia la comunicación (es decir, que inicialmente se pone en contacto con el otro proceso al principio de la sesión) se etiqueta como el cliente. El proceso que espera a ser contactado para comenzar la sesión es el servidor.
   Ej: En una aplicación web, un navegador es un proceso cliente y el servidor web es un proceso servidor. Un ejemplo de la vida real podría ser un empleado de un local de comida rápida que espera pasivamente a los clientes que se acercan a realizar el pedido de su comida.

4) Los agentes de usuario permiten a los usuarios leer, responder, reenviar, guardar y componer mensajes (por ejemplo en los correos electrónicos). Funciona como un intermediario entre el usuario/cliente y el servidor.

5) **HTML:** Es un estándar para los formatos de documentos. 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.
   HTTP: El protocolo de la capa de aplicación de la Web, HTTP, define el formato y la secuencia de los mensajes que se pasan entre el navegador web y el servidor web.

Por tanto, HTTP es sólo una pieza (aunque una pieza importante) de la aplicación web. El programa cliente y el programa servidor, 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.

6)

a) Cuando un usuario solicita una página web, el navegador envía al servidor mensajes de solicitud HTTP para los objetos contenidos en la página. El servidor recibe las solicitudes y responde con mensajes de respuesta HTTP que contienen los objetos.
La primera línea de un mensaje de solicitud HTTP se denomina línea de solicitud y las siguientes líneas son las líneas de cabecera. La línea de solicitud consta de tres campos: el campo de método, el campo URL y el campo de la versión HTTP. El campo que especifica el método puede tomar diferentes valores, entre los que se incluyen GET, POST, HEAD, PUT y DELETE.
Las cabeceras (en inglés headers) HTTP permiten al cliente y al servidor enviar información adicional junto a una petición o respuesta. Una cabecera de petición está compuesta por su nombre (no sensible a las mayúsculas) seguido de dos puntos ':', y a continuación su valor (sin saltos de línea).
La información de la que nos habla es la que está en el encabezado del objeto HTTP, que tiene un método de pedido y un código de respuesta.

b) Una cabecera de petición está compuesta por su nombre (no sensible a las mayúsculas) seguido de dos puntos ':', y a continuación su valor (sin saltos de línea). Los espacios en blanco a la izquierda del valor son ignorados.
Las Cabeceras pueden ser agrupadas de acuerdo a sus contextos:
   - **Cabecera general:** Cabeceras que se aplican tanto a las peticiones como a las respuestas, pero sin relación con los datos que finalmente se transmiten en el cuerpo.
   - **Cabecera de consulta:** Cabeceras que contienen más información sobre el contenido que va a obtenerse o sobre el cliente.
   - **Cabecera de respuesta:** Cabeceras que contienen más información sobre el contenido, como su origen o el servidor (nombre, versión, etc.).
   - **Cabecera de entidad:** Cabeceras que contienen más información sobre el cuerpo de la entidad, como el tamaño del contenido o su tipo MIME.

   Las cabeceras HTTP sirven para transmitir metadatos entre el cliente y el servidor. Estos metadatos incluyen:
   - **Información sobre el mensaje:** Como el tipo de contenido (Content-Type), el tamaño del contenido (Content-Length), y la fecha de modificación (Last-Modified).
   - **Control de caché:** Encabezados como Cache-Control y Expires gestionan la forma en que los recursos deben ser almacenados en caché.

- **Autenticación y autorización:** Encabezados como Authorization en los requerimientos permiten al cliente enviar credenciales, mientras que encabezados como WWW-Authenticate en las respuestas indican al cliente que se requiere autenticación.
- **Redirección:** Encabezados como Location en las respuestas indican al cliente que debe realizar una nueva solicitud a una URL diferente.

El formato de los headers suele ser:

El método - version o version - código de respuesta seguido de:
Nombre-Del-Encabezado: Valor

c) curl -X GET "http://www.misitio.com/index.html" -H "Host: www.misitio.com" -H "User-Agent: curl/7.74.0" -H "Accept: */*"

7) -I: --head Show document info only
   - Este parámetro solicita solo los encabezados HTTP de la respuesta. Es útil para obtener información sobre los metadatos de una respuesta HTTP sin descargar el cuerpo del mensaje. *curl -I http://example.com*

-H: --header <header/@file> Pass custom header(s) to server
   - Este parámetro permite agregar encabezados personalizados a la solicitud HTTP. Puedes usarlo para enviar información adicional, como tokens de autenticación o tipo de contenido. curl -H "Authorization: Bearer YOUR_TOKEN" http://example.com

-X: --request <command> Specify request command to use
--request-target Specify the target for this request
--resolve <host:port:addr[,addr]...> Resolve the host+port to this address
--retry <num> Retry request if transient problems occur
--retry-all-errors Retry all errors (use with --retry)
--retry-connrefused Retry on connection refused (use with --retry)
--retry-delay <seconds> Wait time between retries
--retry-max-time <seconds> Retry only within this period
--sasl-authzid <identity> Identity for SASL PLAIN authentication
--sasl-ir Enable initial response in SASL authentication
--service-name <name> SPNEGO service name
   - Este parámetro especifica el método HTTP a utilizar (por defecto es GET). Puedes usarlo para realizar solicitudes con otros métodos como POST, PUT, DELETE, etc. curl -X POST -d "param1=value1&param2=value2" http://example.com

-s: --silent Silent mode
--socks4 <host[:port]> SOCKS4 proxy on given host + port
--socks4a <host[:port]> SOCKS4a proxy on given host + port
--socks5 <host[:port]> SOCKS5 proxy on given host + port
--socks5-basic Enable username/password auth for SOCKS5 proxies
--socks5-gssapi Enable GSS-API auth for SOCKS5 proxies
--socks5-gssapi-nec Compatibility with NEC SOCKS5 server
--socks5-gssapi-service <name> SOCKS5 proxy service name for GSS-API

--socks5-hostname <host[:port]> SOCKS5 proxy, pass host name to proxy

- Este parámetro activa el modo "silencioso". Suprime la barra de progreso y los mensajes de error, mostrando solo el contenido de la respuesta. Es útil cuando se desea evitar la salida adicional. curl -s http://example.com

8)

a) Un solo requerimiento GET, recibí un archivo HTML con bienvenida a Redes y Comunicaciones.
b) Hacen una solicitud HTTP/S para descargar los recursos, luego se aplican dependiendo del tipo de recurso. El atributo href en la etiqueta <link> se usa para referenciar recursos externos, como hojas de estilo (CSS), iconos (favicon.ico), y fuentes externas. Aunque href no se usa en <img>, el atributo equivalente es src, que especifica la fuente de la imagen. Carga un recurso en la página (imágenes, scripts, videos, etc.)
c) Cuando haces una solicitud GET a una URL, el servidor responde con el HTML principal de la página. Sin embargo, este HTML puede incluir referencias a otros recursos. El navegador analiza el HTML y detecta los archivos referenciados, generando múltiples solicitudes HTTP para cada uno de ellos.
   Ejemplo de tráfico real de una página:
   - GET /index.html → Responde el HTML de la página.
   - GET /styles.css → Descarga el CSS.
   - GET /script.js → Descarga y ejecuta el JavaScript.
   - GET /imagen.jpg → Descarga la imagen.
   - GET /api/data → Obtiene datos desde una API.
d) Serían 8 request, uno para la página, 2 para los archivos CSS, 2 para los archivos JavaScript y 3 para las imágenes.
   Cuando abres la página en un navegador:
   - El navegador realiza un primer request para obtener el HTML.
   - Luego, analiza el HTML y detecta las referencias a archivos externos (CSS, JS, imágenes).
   - Realiza 7 requests adicionales en paralelo para obtener cada recurso referenciado.
   - Una vez descargados, el navegador procesa y renderiza la página con todos los elementos visuales.

   Si ejecutas el comando curl para realizar el request de la página, solo obtendrá el HTML de la página, debes hacer los otros 7 request adicionales para el resto de la información.

9)

a) El primer comando descarga todo el contenido de la respuesta (aunque este se descarta), mientras que el segundo comando solo solicita las cabeceras.
b) El primer comando utiliza > /dev/null para descartar el contenido, mientras que el segundo comando no necesita redirección ya que solo está interesado en las cabeceras.
c) En el segundo comando el requerimiento viajaron 3: host user agent y accept, en la tercera viajaron muchas más: fecha, server, ultima modificación, ETag, accept-ranges, content-length, content-type.

10) Date: Sun, 30 Mar 2025 20:31:24 GMT
La fecha de origen del mensaje.
11) En HTTP/1.0, el cliente sabe que ha recibido todo el objeto solicitado a través del código de estado y el tamaño del contenido. Las principales características son:
    - Código de Estado 200 OK: Indica que la solicitud fue exitosa y que el servidor está enviando la respuesta completa.
    - Content-Length Header: El servidor incluye una cabecera Content-Length en la respuesta que especifica la longitud del cuerpo del mensaje en bytes.

    HTTP/1.1, Además de Content-Length, HTTP/1.1 introduce Transfer-Encoding: chunked para permitir la transferencia en fragmentos. El cliente detecta el final del objeto cuando recibe un fragmento de tamaño cero.

12) Los códigos de estado HTTP son respuestas que un servidor web devuelve para indicar el resultado de una solicitud. Estos códigos se clasifican en cinco categorías principales, representadas por el primer dígito del código.
    1XX - Respuestas Informativas: Indican que el servidor ha recibido la solicitud y que el cliente debe esperar antes de completar la transacción.
    - **100 Continue** → El servidor recibió la solicitud y el cliente puede seguir enviando el resto de los datos.
    - **101 Switching Protocols** → El servidor acepta cambiar a un protocolo diferente solicitado por el cliente.

    2XX - Respuestas Exitosas: Indican que la solicitud del cliente fue recibida, entendida y procesada correctamente.
    - **200 OK** → La solicitud fue exitosa y el servidor devuelve el contenido solicitado.
    - **201 Created** → Se ha creado un nuevo recurso en el servidor.
    - **204 No Content** → La solicitud fue exitosa, pero no hay contenido en la respuesta.

    3XX - Redirecciones: Indican que el cliente debe realizar otra acción para completar la solicitud.
    - **301 Moved Permanently** → La URL solicitada se ha movido de forma permanente a otra ubicación.
    - **302 Found** → La URL ha sido encontrada, pero se debe hacer la solicitud en una nueva ubicación temporalmente.
    - **304 Not Modified** → El recurso no ha cambiado, por lo que el navegador puede usar su versión en caché.

    4XX - Errores del Cliente: Indican que hay un problema en la solicitud enviada por el cliente (malformada, falta de permisos, recurso inexistente, etc.).
    - **400 Bad Request** → La solicitud es incorrecta o está mal estructurada.
    - **401 Unauthorized** → Se requiere autenticación para acceder al recurso.
    - **403 Forbidden** → El acceso al recurso está prohibido.
    - **404 Not Found** → El recurso solicitado no existe en el servidor.
    - **405 Method Not Allowed** → El método HTTP utilizado no está permitido para el recurso.

    5XX - Errores del Servidor: Indican que el servidor falló al procesar una solicitud válida debido a un problema interno.

- **500 Internal Server Error** → Error genérico del servidor.
- **502 Bad Gateway** → El servidor actuó como puerta de enlace y recibió una respuesta inválida de otro servidor.
- **503 Service Unavailable** → El servidor está sobrecargado o en mantenimiento.
- **504 Gateway Timeout** → El servidor no recibió respuesta a tiempo desde otro servidor.

13)

a) HTTP/1.1 200 OK. Indica la versión HTTP y la respuesta, en este caso, exitosa.
b) 7
c) Server: Apache/2.4.56 (Unix)
d) SÍ
e) Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
f) curl -I GET "http://www.redes.unlp.edu.ar" -H "If-Modified-Since: Sun, 19 Mar 2023 19:04:46 GMT"
   - curl: (6) Could not resolve host: GET
   - HTTP/1.1 304 Not Modified

14) curl GET "http://www.redes.unlp.edu.ar/restringido/index.php"
- curl: (6) Could not resolve host: GET
- <h1>Acceso restringido</h1>
- 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

curl GET "http://www.redes.unlp.edu.ar/obtener-usuario.php"
- curl: (6) Could not resolve host: GET
- Para obtener el usuario y la contraseña haga un requerimiento a esta página seteando el encabezado 'Usuario-Redes' con el valor 'obtener'

curl GET "http://www.redes.unlp.edu.ar/obtener-usuario.php" -H "Usuario-Redes: obtener"
- curl: (6) Could not resolve host: GET
- 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!

echo -n "redes:RYC" | base64
- cmVkZXM6UllD

curl GET "http://www.redes.unlp.edu.ar/restringido/index.php" -H "Authorization: Basic cmVkZXM6UllD"
- curl: (6) Could not resolve host: GET
- <h1>Excelente!</h1>
- Para terminar el ejercicio deberás agregar en la entrega los datos que se muestran en la siguiente página.

- ACLARACIÓN: la URL de la siguiente página está contenida en esta misma respuesta.

curl -H "Authorization: Basic cmVkZXM6UlID" -I www.redes.unlp.edu.ar/restringido/index.php

- HTTP/1.1 302 Found
- Date: Thu, 03 Apr 2025 12:00:43 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-Type: text/html; charset=UTF-8

curl -H "Authorization: Basic cmVkZXM6UlID" www.redes.unlp.edu.ar/restringido/the-end.php

- ¡Felicitaciones, llegaste al final del ejercicio!
- Fecha: 2025-04-03 12:02:42
- Verificación: 606c3f969453473258ef7c8d5c5e67ce8ec8f77af94453f23d1c3f2c9728ab30

15)

a) curl www.redes.unlp.edu.ar/extras/prueba-http-1-0.txt
   - GET /http/HTTP-1.1/ HTTP/1.0
   - User-Agent: curl/7.38.0
   - Host: www.redes.unlp.edu.ar
   - Accept: */*

b) 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: Thu, 03 Apr 2025 12:13:02 GMT
   Server: Apache/2.4.56 (Unix)
   Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
   ETag: "760-5f7457bd64f80"
   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 Inform&aacute;tica</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 p&aacute;gina se visualiza utilizando HTTP 1.1. Utilizando el
capturador de paquetes analice cuantos flujos utiliza el navegador para
visualizar la p&aacute;gina con sus im&aacute;genes en
contraposici&oacute;n con el protocolo HTTP/1.0.
  </p>
  </p>
  <h2>Imagen de ejemplo</h2>
```

```
    <img src="13532-tuxkiller03green.png" width="800px"/>
    </div>


    </div>
    <div id="footer">
      <div class="container">
        <p class="muted credit">Redes y Comunicaciones</p>
      </div>
    </div>
  </body>
</html>
Connection closed by foreign host.
```

c)
```
Trying 172.28.0.50...
Connected to www.redes.unlp.edu.ar.
Escape character is '^]'.
GET /http/HTTP-1.1/ HTTP/1.1
User-Agent: curl/7.38.0
Host: www.redes.unlp.edu.ar
Accept: */*


HTTP/1.1 200 OK
Date: Thu, 03 Apr 2025 12:30:58 GMT
Server: Apache/2.4.56 (Unix)
Last-Modified: Sun, 19 Mar 2023 19:04:46 GMT
ETag: "760-5f7457bd64f80"
Accept-Ranges: bytes
Content-Length: 1888
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 Inform&aacute;tica</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 p&aacute;gina se visualiza utilizando HTTP 1.1. Utilizando el
capturador de paquetes analice cuantos flujos utiliza el navegador para
visualizar la p&aacute;gina con sus im&aacute;genes en
contraposici&oacute;n con el protocolo HTTP/1.0.
  </p>
  </p>
  <h2>Imagen de ejemplo</h2>
  <img src="13532-tuxkiller03green.png" width="800px"/>
  </div>



  </div>
  <div id="footer">
   <div class="container">
    <p class="muted credit">Redes y Comunicaciones</p>
   </div>
  </div>
 </body>
</html>
Connection closed by foreign host.
```

16)

a) El comando telnet se usa para establecer una conexión a un servidor remoto utilizando el protocolo Telnet. El comando telnet usa el protocolo Telnet, que es un protocolo de comunicación de texto en TCP/IP. Esta conexión se establece en el puerto 80.
b) GET, se solicitó el html de la página.
c) El primero queda abierto y el segundo no.
d) El caso más eficiente es HTTP/1.1. Esto se debe a que HTTP/1.1 permite el uso de conexiones persistentes, lo que reduce la necesidad de abrir y cerrar conexiones TCP para cada recurso. Esto resulta en menos sobrecarga y tiempos de respuesta más rápidos, especialmente cuando se trata de páginas web que requieren múltiples solicitudes para obtener todos los recursos necesarios (estilos, scripts, imágenes, etc.).

17)

a) <form method="GET" action="metodos-lectura-valores.php">\n
b) <form method="POST" action="metodos-lectura-valores.php">\n
c) La diferencia es que con el método GET la información del formulario viaja en la URL, mientras que con el método POST la información viaja en el cuerpo de la solicitud.
d) Se puede observar en el browser que con el método GET podemos ver la información ingresada en el formulario en los parámetros de la URL.

18) **Cabecera Set-Cookie**

Uso: La cabecera Set-Cookie es enviada por el servidor al cliente (navegador) para establecer una cookie. Esta cookie puede almacenar información como identificadores de sesión, preferencias del usuario, o datos temporales.
Formato: La cabecera tiene la siguiente estructura:
Set-Cookie: nombre=valor; Expires=fecha; Path=/; Domain=ejemplo.com; Secure; HttpOnly; SameSite=Strict

- nombre=valor: Define el nombre y el valor de la cookie.
- Expires: Indica la fecha de caducidad de la cookie.
- Path: Define el ámbito de la cookie, es decir, las rutas en el servidor donde la cookie es válida.
- Domain: Especifica el dominio para el cual la cookie es válida.
- Secure: Indica que la cookie solo debe ser enviada a través de conexiones HTTPS.
- HttpOnly: Evita que la cookie sea accesible a través de JavaScript, ayudando a prevenir ataques XSS (Cross-Site Scripting).
- SameSite: Controla si la cookie se envía en solicitudes cross-site, ayudando a prevenir ataques CSRF (Cross-Site Request Forgery).

Cabecera Cookie

Uso: La cabecera Cookie es enviada por el cliente al servidor en las solicitudes HTTP subsiguientes. Contiene las cookies que el servidor estableció previamente mediante Set-Cookie y que son relevantes para la solicitud actual.
Formato: La cabecera tiene la siguiente estructura:

Cookie: nombre1=valor1; nombre2=valor2
La cabecera lista todas las cookies que el cliente ha almacenado y que son pertinentes para el dominio y la ruta especificados.

19) **Protocolos Basados en Texto**
- Representación: Los protocolos basados en texto transmiten la información en formato legible por humanos. Utilizan caracteres alfanuméricos y otros símbolos para representar datos y comandos.
- El formato textual es fácil de entender y depurar, pero puede ser menos eficiente en términos de tamaño y velocidad de procesamiento en comparación con los protocolos binarios.

Protocolos Binarios
- Representación: Los protocolos binarios transmiten la información en formato binario, es decir, en una secuencia de bytes. Estos protocolos pueden ser más compactos y eficientes en términos de procesamiento y tamaño de los mensajes porque no requieren la conversión entre texto y datos binarios.
- Ejemplo: HTTP/2 es un protocolo binario. En HTTP/2, tanto las solicitudes como las respuestas se codifican en un formato binario, lo que permite una mayor eficiencia en la transmisión de datos y una mejor gestión de múltiples solicitudes y respuestas en una sola conexión.

HTTP/1.0, HTTP/1.1 y HTTP/2
- HTTP/1.0: Es un protocolo basado en texto. La comunicación entre el cliente y el servidor se realiza mediante mensajes de texto, con encabezados y cuerpos de mensaje en formato legible por humanos.
- HTTP/1.1: También es un protocolo basado en texto, aunque introdujo varias mejoras respecto a HTTP/1.0, como el soporte para conexiones persistentes y pipelining de solicitudes.
- HTTP/2: Es un protocolo binario. Introdujo un nuevo formato de codificación binario para solicitudes y respuestas, mejorando la eficiencia en la transmisión de datos y permitiendo características avanzadas como multiplexación de flujos, compresión de encabezados y gestión eficiente de conexiones.

20)

a) HTTP/1.1: La cabecera Host es obligatoria y permite identificar el dominio en servidores que usan alojamiento virtual.

   HTTP/1.0: No era obligatorio.

   HTTP/2: La cabecera Host sigue siendo necesaria y se utiliza para la misma finalidad: especificar el nombre del host y el puerto. HTTP/2 se basa en el mismo concepto de virtual hosting que HTTP/1.1. Sin embargo, HTTP/2 utiliza una compresión de cabeceras (HPACK) para reducir la cantidad de datos que se envían en las solicitudes y respuestas, lo que puede reducir la redundancia y mejorar la eficiencia en la transmisión de datos.

b) Si es correcto.

c) :method: GET
:scheme: https
:authority: www.info.unlp.edu.ar
:path: /index.php

Ejercicio Parcial:

a) 1.0 o superior basado en los headers, pero la conexión se cierra después de la request así que tal vez sea la 1.0 específicamente.
b) HEAD, Si retorna todo el recurso solicitado.
c) l/métodos/
d) Si, código 200 ok.
e) Hubiese retornado un 304 y el navegador usará la versión cacheada si hay.