1. 1**¿Cuál es la función de la capa de aplicación?**
   * Proveer servicios de comunicación a los usuarios y a las aplicaciones, incluye las aplicaciones mismas. Puedo ayudarte a revisar el contenido, resumir documentos, generar borradores de correos electrónicos, documentos y más. Si necesitas asistencia con algo específico, por favor házmelo saber.
   * Modelo de comunicación machine to machine (M2M), no hay usuarios (personas).
   * Las aplicaciones que usan la red pertenecen a esta capa.
   * Los protocolos que implementan las aplicaciones también.
   * Existen aplicaciones que NO son de red que deben trabajar con aplicaciones/servicios para lograr acceso a la red.
2. **Si dos procesos deben comunicarse:**
   * **a. ¿Cómo podrían hacerlo si están en diferentes máquinas?**
     + Se le indica el número de puerto sobre la URL para ser conectado.
   * **b. Y si están en la misma máquina, ¿qué alternativas existen?**
     + Se comunicarían por su identificador único.
3. **Explique brevemente cómo es el modelo Cliente/Servidor. Dé 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?**
   * El modelo Cliente/Servidor pone al usuario el procesamiento de interfaz y al server el resto del procesamiento.
   * El servidor corre de forma pasiva esperando la conexión.
   * El cliente se conecta al server y de esta forma se comunican.
   * Un ejemplo sería un cajero automático o ATM, el cual está constantemente escuchando y el cliente se conecta para interactuar.
   * Otros ejemplos de comunicación de aplicaciones: Modelo Mainframe (dumb client), Modelo Cliente/Servidor, Modelo Peer to Peer (P2P), Modelo Híbrido.
4. **Describa la funcionalidad de la entidad genérica “Agente de usuario” o “User agent”.**
5. **¿Qué son y en qué se diferencian HTML y HTTP?**
   * HTTP es un protocolo de capa de aplicación que sirve para comunicar servicios.
   * HTTP 1.0 no hay conexión persistente, en cambio HTTP 1.1 se arma una conexión TCP y se pueden hacer varias peticiones.
   * Un petición HTTP puede ser el comando curl que actuaría como un GET.
   * Un recurso que se solicita podría ser el index.html por ejemplo.
   * Cuando haces un GET / se lo haces a un objeto.
   * Si en una página web hay 4 “HREF”, se cuenta 1 petición HTTP por cada HREF y una más por la página, ósea, 5 Peticiones HTTP.
   * Para contar las peticiones web a través de un curl hay que contar las etiquetas html .
6. **HTTP tiene definido un formato de mensaje para los requerimientos y las respuestas. (Ayuda: apartado “Formato de mensaje HTTP”, Kurose).**
   * **a. ¿Qué información de la capa de aplicación nos indica si un mensaje es de requerimiento o de respuesta para HTTP? ¿Cómo está compuesta dicha información? ¿Para qué sirven las cabeceras?**
   * **b. ¿Cuál es su formato? (Ayuda:([https://developer.mozilla.org/es/docs/Web/HTTP/Headers](https://developer.mozilla.org/es/docs/Web/HTTP/Headers)))**
   * **c. Suponga que desea enviar un requerimiento con la versión de HTTP 1.1 desde curl/7.74.0 a un sitio de ejemplo como [www.misitio.com](http://www.misitio.com) para obtener el recurso /index.html. En base a lo indicado, ¿qué información debería enviarse mediante encabezados? Indique cómo quedaría el requerimiento.**
7. **Utilizando la VM, abra una terminal e investigue sobre el comando curl. Analice para qué sirven los siguientes parámetros (-I, -H, -X, -s).**
   * El “-i” Este parámetro incluye los encabezados de la respuesta HTTP en la salida del comando. Es útil para ver información adicional, como códigos de estado, cookies, y otros metadatos.
   * El “-h” Este se utiliza para especificar encabezados personalizados en la solicitud. Por ejemplo, si quieres enviar un encabezado Authorization o Content-Type.
   * El “-x” Permite especificar un servidor proxy para enviar la solicitud. Es útil si deseas enrutar tu solicitud HTTP a través de un proxy.
   * El “-s” Suprime la salida de información como detalles del progreso y errores. Solo devuelve la salida principal de la solicitud, lo que es útil para scripts o cuando quieres mantener la consola más limpia.
8. **Ejecute el comando curl sin ningún parámetro adicional y acceda a [www.redes.unlp.edu.ar](http://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.**
   * **b. ¿Cómo funcionan los atributos href de los tags link e img en html?**
   * **c. Para visualizar la página completa con imágenes como en un navegador, ¿alcanza con realizar un único requerimiento?**
   * **d. ¿Cuántos requerimientos serían necesarios para obtener una página que tiene dos CSS, dos Javascript y tres imágenes? Diferencie cómo funcionaría un navegador respecto al comando curl ejecutado previamente.**
9. **Ejecute a continuación los siguientes comandos:**
   * **curl -v -s [www.redes.unlp.edu.ar](http://www.redes.unlp.edu.ar) > /dev/null**
   * **curl -I -v -s [www.redes.unlp.edu.ar](<http://www.redes.unlp.edu.ar>)**
   * **a. ¿Qué diferencias nota entre cada uno?**
     + Primero (-v -s): Descarga toda la respuesta HTTP (cabeceras y cuerpo). Sin embargo, como la salida del cuerpo se redirige a /dev/null, solo verás información de conexión y cabeceras mostradas por -v.
     + Segundo (-l -v -s): Solo solicita y muestra las cabeceras de la respuesta HTTP. No descarga el cuerpo, lo que hace que sea más rápido y no requiere redirección para evitar mostrar datos innecesarios.
   * **b. ¿Qué ocurre si en el primer comando se quita la redirección a /dev/null? ¿Por qué no es necesaria en el segundo comando?**
     + Sin redirección (/dev/null): El cuerpo de la respuesta (contenido de la página web) se mostrará en la salida estándar junto con los mensajes verbosos. Esto puede generar mucho ruido en la terminal, especialmente si la página contiene contenido grande o no textual.
     + Por qué no es necesario en el segundo comando: En el caso del segundo comando, “-l” asegura que solo se descarguen y muestren las cabeceras, así que no hay cuerpo que redirigir. Esto hace innecesaria la redirección.
   * **c. ¿Cuántas cabeceras viajaron en el requerimiento? ¿Y en la respuesta?**
     + En la petición viajaron 3
     + En la respuesta viajaron 7
10. **¿Qué indica la cabecera Date?**
    * La cabecera “Date” en una respuesta HTTP indica la fecha y hora en que el servidor generó la respuesta. Es una referencia temporal que sigue el formato del estándar RFC 7231.
11. **En HTTP/1.0, ¿cómo sabe el cliente que ya recibió todo el objeto solicitado de manera completa? ¿Y en HTTP/1.1?**
    * En HTTP 1.1 el servidor incluye la cabecera “Content Type” en la respuesta para indicar explícitamente el tamaño del contenido en bytes. El cliente sabe que ha recibido todo el objeto cuando alcanza esa cantidad de datos.
    * En HTTP 1.0 el cliente utiliza el cierre de la conexión TCP como señal para determinar que ha recibido todo el contenido. Esto ocurre ya que la señal no es persistente como en el 1.1
12. **Investigue los distintos tipos de códigos de retorno de un servidor web y su significado. Considere que los mismos se clasifican en categorías (2XX, 3XX, 4XX, 5XX).**
    * **1XX - Respuestas Informativas**

Indican que la solicitud fue recibida y el proceso continúa:

* + - **100 Continue:** El cliente puede continuar con su solicitud.
    - **101 Switching Protocols:** El servidor acepta cambiar el protocolo (p. ej., de HTTP a WebSocket).
    - **103 Early Hints:** Sugiere recursos que pueden precargarse antes de la respuesta final.
  + **2XX - Respuestas Satisfactorias**

La solicitud fue recibida, comprendida y procesada correctamente:

* + - **200 OK:** La solicitud fue exitosa y el resultado se devuelve.
    - **201 Created:** Un recurso fue creado exitosamente (común en POST).
    - **202 Accepted:** La solicitud fue recibida pero no procesada aún.
    - **204 No Content:** La solicitud fue exitosa, pero no hay contenido que devolver.
    - **206 Partial Content:** Respuesta parcial por solicitudes de rango (útil para descargas).
  + **3XX - Redirecciones**

El cliente debe realizar acciones adicionales para completar la solicitud:

* + - **301 Moved Permanently:** La URL ha cambiado permanentemente.
    - **302 Found (Temporalmente movido):** La URL ha cambiado temporalmente.
    - **303 See Other:** Redirección a una URL distinta (tras un POST).
    - **304 Not Modified:** El contenido no ha cambiado; se puede usar el caché.
    - **307 Temporary Redirect:** Igual que 302, pero el método HTTP no cambia.
    - **308 Permanent Redirect:** Igual que 301, pero el método HTTP no cambia.
  + **4XX - Errores del Cliente**

El cliente hizo una solicitud incorrecta:

* + - **400 Bad Request:** Solicitud malformada o incorrecta.
    - **401 Unauthorized:** Autenticación requerida o fallida.
    - **403 Forbidden:** Acceso prohibido (autenticación válida, pero sin permisos).
    - **404 Not Found:** Recurso no encontrado.
    - **405 Method Not Allowed:** Método HTTP no soportado.
    - **408 Request Timeout:** El cliente tardó demasiado en enviar la solicitud.
    - **429 Too Many Requests:** Exceso de solicitudes en un tiempo breve.
  + **5XX - Errores del Servidor**

El servidor falló al procesar la solicitud:

* + - **500 Internal Server Error:** Error genérico del servidor.
    - **501 Not Implemented:** Funcionalidad no soportada por el servidor.
    - **502 Bad Gateway:** Respuesta inválida desde un servidor intermedio.
    - **503 Service Unavailable:** El servicio no está disponible temporalmente.
    - **504 Gateway Timeout:** Un servidor intermedio no respondió a tiempo.
    - **505 HTTP Version Not Supported:** Versión de HTTP no soportada.

1. **Utilizando curl, realice un requerimiento con el método HEAD al sitio [www.redes.unlp.edu.ar](http://www.redes.unlp.edu.ar) e indique:**

* **a. ¿Qué información brinda la primera línea de la respuesta?**
* **b. ¿Cuántos encabezados muestra la respuesta?**
* **c. ¿Qué servidor web está sirviendo la página?**
* **d. ¿El acceso a la página solicitada fue exitoso o no?**
* **e. ¿Cuándo fue la última vez que se modificó la página?**
* **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?**

1. **Utilizando curl, acceda al sitio [www.redes.unlp.edu.ar/restringido/index.php](http://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.**
2. **Utilizando la VM, realice las siguientes pruebas:**

* **a. Ejecute el comando ’curl [www.redes.unlp.edu.ar/extras/prueba-http-1-0.txt’](http://www.redes.unlp.edu.ar/extras/prueba-http-1-0.txt%E2%80%99) 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](http://www.redes.unlp.edu.ar) 80 y luego pegue el contenido que tiene almacenado en el portapapeles. ¿Qué ocurre luego de hacerlo?**
* **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 el comando telnet nuevamente.**

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

* **a. ¿Qué está haciendo al ejecutar el comando telnet?**
* **b. ¿Qué método HTTP utilizó? ¿Qué recurso solicitó?**
* **c. ¿Qué diferencias notó entre los dos casos? ¿Puede explicar por qué?**
* **d. ¿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?**

1. **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 menú “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](http://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:**
* **b. Analice el código HTML**
* **c. Utilizando el analizador de paquetes Wireshark capture los paquetes enviados y recibidos al presionar el botón Enviar.**
* **d. ¿Qué diferencias detectó en los mensajes enviados por el cliente?**
* **e. ¿Observó alguna diferencia en el browser si se utiliza un mensaje u otro?**

1. **Investigue cuál es el principal uso que se le da a las cabeceras Set-Cookie y Cookie en HTTP y qué relación tienen con el funcionamiento del protocolo HTTP.**
2. **¿Cuál es la diferencia entre un protocolo binario y uno basado en texto? ¿De qué tipo de protocolo se trata HTTP/1.0, HTTP/1.1 y HTTP/2?**
3. **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](https://undertow.io/blog/2015/04/27/An-in-depth-overview-of-HTTP2.html)) para HTTP/2)**
* **b. En HTTP/1.1, ¿es correcto el siguiente requerimiento?**
  + **GET /index.php HTTP/1.1**
  + **User-Agent: curl/7.54.0**
* **c. ¿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](http://www.info.unlp.edu.ar)**
* **Ejercicio de Parcial**
  + **curl -X ?? [www.redes.unlp.edu.ar/](http://www.redes.unlp.edu.ar/)??**
  + **> HEAD /metodos/ HTTP/??**
  + **> Host: [www.redes.unlp.edu.ar](http://www.redes.unlp.edu.ar)**
  + **> User-Agent: curl/7.54.0**
  + **< HTTP/?? 200 OK**
  + **< Server: nginx/1.4.6 (Ubuntu)**
  + **< Date: Wed, 31 Jan 2018 22:22:22 GMT**
  + **< Last-Modified: Sat, 20 Jan 2018 13:02:41 GMT**
  + **< Content-Type: text/html; charset=UTF-8**
  + **< Connection: close**
* **a. ¿Qué versión de HTTP podría estar utilizando el servidor?**
* **b. ¿Qué método está utilizando? Dicho método, ¿retorna el recurso completo solicitado?**
* **c. ¿Cuál es el recurso solicitado?**
* **d. ¿El método funcionó correctamente?**
* **e. Si la solicitud hubiera llevado un encabezado que diga:**
  + **If-Modified-Since: Sat, 20 Jan 2018 13:02:41 GMT**
  + **¿Cuál habría sido la respuesta del servidor web? ¿Qué habría hecho el navegador en este caso?**