TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLOGICO NUEVO LAREDO

Miguel Angel Dávila Sánchez (22100177)

Docente:

Gerardo Espinoza Zapata

Programacion web

Investigación http

1er Parcial

27 de septiembre 2024

27/09/2024 Ingeniería en Sistemas

Índice

1. Protocolos de Internet 3
2. Protocolo HTTP y HTTPS 5
3. Protocolo SSL/TLS 6
4. Partes de una Petición y Respuesta HTTP 6
5. Encabezados en HTTP 9
6. Clasificación de Códigos de Estado HTTP 10

1. Protocolos de internet:

Protocolo	Descripción	Uso Común
TCP/IP	Transmission Control Protocol / Internet Protocol	Base para la comunicación en redes de internet. Maneja el direccionamiento y la transferencia de datos. Utilizado en casi todas las comunicaciones de internet.
HTTP	Hypertext Transfer Protocol	Transmisión de documentos de hipertexto (páginas web). Utilizado para la navegación web.
HTTPS	Hypertext Transfer Protocol Secure	Versión segura de HTTP, utilizando SSL/TLS para encriptar la información. Usado para navegación segura en la web.
FTP	File Transfer Protocol	Transferencia de archivos entre sistemas en red. Utilizado para cargar y descargar archivos a través de internet.
SMTP	Simple Mail Transfer Protocol	Envía correos electrónicos entre servidores de correo electrónico. Utilizado en servicios de correo electrónico.
IMAP	Internet Message Access Protocol	Acceso a correos electrónicos almacenados en un servidor. Permite gestionar el correo electrónico de manera más eficiente.
POP3	Post Office Protocol version 3	Descarga correos electrónicos desde un servidor al dispositivo local.

		Común en clientes de correo electrónico.
DNS	Domain Name System	Traduce nombres de dominio (como www.example.com) a direcciones IP. Esencial para la navegación web.
SSH	Secure Shell	Acceso remoto seguro a sistemas en red. Utilizado para administración de servidores y transferencia segura de datos.
TELNET	Telnet Protocol	Protocolo de acceso remoto a sistemas, no seguro. Reemplazado por SSH en la mayoría de los casos.
UDP	User Datagram Protocol	Protocolo sin conexión que envía datagramas sin confirmación de entrega. Usado en streaming y juegos en línea.
SFTP	SSH File Transfer Protocol	Transferencia segura de archivos, basado en SSH. Alternativa segura a FTP.
DHCP	Dynamic Host Configuration Protocol	Asigna dinámicamente direcciones IP a dispositivos en una red. Facilita la administración de direcciones IP.
ICMP	Internet Control Message Protocol	Envía mensajes de control y error en redes. Utilizado en herramientas de diagnóstico como "ping".
ARP	Address Resolution Protocol	Resuelve direcciones IP a direcciones MAC en una red local. Esencial para la comunicación en redes LAN.

2. Protocolo http y https

HTTP es el protocolo de comunicación principal utilizado para la transferencia de documentos de hipertexto en la World Wide Web. Se utiliza para intercambiar información entre un cliente (normalmente un navegador web) y un servidor web.

Características de HTTP:

- Transmisión de Datos: HTTP permite la transferencia de archivos como texto, imágenes, videos y otros tipos de datos. Cada vez que visitas una página web, tu navegador utiliza HTTP para solicitar el contenido de esa página desde el servidor.
- Modelo Cliente-Servidor: Funciona en un modelo cliente-servidor, donde el cliente envía una solicitud al servidor, y el servidor responde con el recurso solicitado.
- Sin Estado (Stateless): HTTP es un protocolo sin estado, lo que significa que cada solicitud y respuesta son independientes y el protocolo no retiene información sobre sesiones anteriores.

Usos Comunes:

- Navegación web.
- Transferencia de datos entre aplicaciones web y servidores.
- Comunicación con APIs (interfaces de programación de aplicaciones).

Protocolo HTTPS (Hypertext Transfer Protocol Secure)

HTTPS es una versión segura de HTTP que utiliza criptografía para proteger la integridad y confidencialidad de los datos transmitidos entre el cliente y el servidor.

Características de HTTPS:

- Seguridad: HTTPS añade una capa de seguridad a HTTP utilizando el protocolo SSL/TLS (Secure Sockets Layer / Transport Layer Security) para encriptar la información transmitida.
- Confidencialidad: La encriptación de los datos asegura que la información intercambiada no pueda ser interceptada y leída por terceros, protegiendo así la privacidad del usuario.
- Integridad: HTTPS garantiza que los datos no sean alterados durante la transmisión, asegurando que el contenido recibido por el usuario sea el mismo que el enviado por el servidor.
- Autenticación: HTTPS verifica la autenticidad del servidor, asegurándose de que el cliente está comunicándose con el servidor legítimo y no con un impostor.

Usos Comunes:

- Navegación segura en sitios web (indicados con un candado en la barra de direcciones del navegador).
- Transacciones financieras en línea, como pagos y transferencias bancarias.
- Envío de información sensible, como contraseñas y datos personales.

Que es o para que nos sirve el protocolo SSL/TLS

Protocolo SSL/TLS (Secure Sockets Layer / Transport Layer Security)

SSL y TLS son protocolos criptográficos que proporcionan seguridad para las comunicaciones en la red. Aunque SSL fue el precursor, TLS es su sucesor y es más seguro y eficiente. Ambos se utilizan para proteger la transmisión de datos en aplicaciones como HTTPS, correo electrónico, mensajería instantánea y más.

¿Qué es SSL/TLS?

- SSL (Secure Sockets Layer): Fue desarrollado por Netscape en la década de 1990 como un protocolo para establecer un canal seguro entre el cliente y el servidor. Las versiones anteriores (SSL 2.0 y SSL 3.0) se consideran inseguras y han sido reemplazadas por TLS.
- TLS (Transport Layer Security): Es la evolución de SSL. Mejora la seguridad y la eficiencia del protocolo original. Las versiones actuales de TLS (1.2 y 1.3) son los estándares recomendados para la seguridad de las comunicaciones.

¿Para qué sirve SSL/TLS?

- Encriptación de Datos: Protege la privacidad de los datos transmitidos en la red mediante la encriptación, evitando que terceros puedan leer la información durante la transmisión.
- Autenticación: Verifica la identidad del servidor (y opcionalmente del cliente) mediante certificados digitales, asegurando que el cliente se comunica con el servidor legítimo.
- Integridad de los Datos: Utiliza firmas digitales para asegurar que los datos no hayan sido modificados durante el tránsito, garantizando que el contenido recibido es auténtico.

3. Cuáles son las partes de una petición y una respuesta HTTP

Partes de una Petición HTTP

Una petición HTTP es enviada por el cliente al servidor para solicitar un recurso específico, como una página web o un archivo. La estructura de la petición consta de las siguientes partes:

Línea de Petición (Request Line)

- Formato: Método HTTP + Ruta del recurso + Versión del Protocolo
- Ejemplo: GET /index.html HTTP/1.1
- Elementos:
    - Método HTTP: Define la acción que se desea realizar sobre el recurso. Los métodos más comunes son:
        - GET: Solicita un recurso sin modificarlo.
        - POST: Envía datos al servidor para procesarlos (como un formulario).
        - PUT: Actualiza o crea un recurso en una ubicación específica.
        - DELETE: Elimina un recurso.
        - HEAD: Igual que GET, pero solo solicita los encabezados, no el cuerpo del recurso.
    - Ruta del Recurso (URI): Indica la ubicación del recurso en el servidor. Puede ser una ruta relativa (/index.html) o un URI completo (http://www.example.com/index.html).
    - Versión del Protocolo: Especifica la versión de HTTP que se está utilizando, como HTTP/1.1 o HTTP/2.

Encabezados de Petición (Request Headers)

- Los encabezados proporcionan información adicional sobre la petición o sobre el cliente que la realiza. Cada encabezado consta de un nombre seguido de un valor.
- Ejemplos Comunes:
    - Host: Especifica el dominio del servidor (por ejemplo, Host: www.example.com).
    - User-Agent: Identifica el cliente que realiza la petición (por ejemplo, el navegador y su versión).
    - Accept: Indica los tipos de contenido que el cliente acepta (por ejemplo, text/html, application/json).
    - Accept-Language: Especifica el idioma preferido para la respuesta (por ejemplo, en-US).
    - Content-Type: Indica el tipo de contenido enviado en el cuerpo de la petición (si lo hay), como application/json o text/html.
    - Authorization: Contiene credenciales para autenticarse en el servidor, como tokens o credenciales básicas.

Cuerpo de la Petición (Request Body)

- El cuerpo de la petición contiene los datos que se envían al servidor. No todas las peticiones tienen cuerpo; típicamente lo tienen las peticiones POST, PUT y PATCH.
- Ejemplos:
    - En un formulario HTML enviado con POST, el cuerpo contiene los datos del formulario.
    - En una API RESTful, el cuerpo de una petición POST o PUT puede contener un objeto JSON que representa datos a guardar o actualizar.

Partes de una Respuesta HTTP

Una respuesta HTTP es enviada por el servidor al cliente en respuesta a una petición. Contiene información sobre el estado de la solicitud y el recurso solicitado. La estructura de la respuesta consta de las siguientes partes:

Línea de Respuesta (Status Line)

- Formato: Versión del Protocolo + Código de Estado + Mensaje de Estado
- Ejemplo: HTTP/1.1 200 OK
- Elementos:
    - Versión del Protocolo: Indica la versión de HTTP utilizada en la respuesta, como HTTP/1.1 o HTTP/2.
    - Código de Estado: Es un número que indica el resultado de la petición:
        - 1xx: Informacional (procesando).
        - 2xx: Éxito (por ejemplo, 200 OK).
        - 3xx: Redirección (por ejemplo, 301 Moved Permanently).
        - 4xx: Error del cliente (por ejemplo, 404 Not Found).
        - 5xx: Error del servidor (por ejemplo, 500 Internal Server Error).
    - Mensaje de Estado: Texto descriptivo que acompaña al código de estado (por ejemplo, OK, Not Found, Internal Server Error).

Encabezados de Respuesta (Response Headers)

- Los encabezados de respuesta proporcionan información adicional sobre la respuesta o sobre el propio servidor. Al igual que en las peticiones, cada encabezado consta de un nombre seguido de un valor.
- Ejemplos Comunes:
    - Content-Type: Indica el tipo de contenido del cuerpo de la respuesta (por ejemplo, text/html, application/json).

- Content-Length: Especifica la longitud del contenido en bytes.
- Set-Cookie: Envía cookies al cliente para su almacenamiento.
- Cache-Control: Instrucciones sobre el almacenamiento en caché de la respuesta.
- Server: Informa sobre el software del servidor que genera la respuesta (por ejemplo, Apache/2.4.1).

Cuerpo de la Respuesta (Response Body)

- El cuerpo de la respuesta contiene los datos solicitados. Puede ser una página HTML, un archivo, una imagen, un JSON, entre otros.
- Ejemplos:
  - Para una petición GET de una página web, el cuerpo de la respuesta contiene el código HTML de la página solicitada.
  - Para una API que devuelve datos, el cuerpo de la respuesta puede contener un objeto JSON con la información solicitada.

4. Que son los encabezados <Headers> en una petición/respuesta HTTP

Los encabezados HTTP son líneas adicionales en las peticiones y respuestas HTTP que contienen información meta acerca de los datos que se están enviando o recibiendo. Se encuentran justo después de la línea de petición o respuesta y antes del cuerpo de la solicitud o respuesta (si es que hay uno).

Los encabezados HTTP cumplen varias funciones, incluyendo:

1. Proporcionar Información Adicional sobre la Petición o Respuesta:
   - Los encabezados informan al servidor o al cliente sobre cómo deben tratar la solicitud o respuesta. Por ejemplo, el encabezado Content-Type indica el tipo de datos que se están enviando o recibiendo (por ejemplo, application/json o text/html).
2. Controlar el Comportamiento de la Comunicación:
   - Los encabezados se utilizan para modificar la forma en que se envían y reciben los datos. Por ejemplo, el encabezado Cache-Control indica si la respuesta puede ser almacenada en caché y por cuánto tiempo.
3. Autenticación y Seguridad:
   - Los encabezados como Authorization permiten enviar credenciales de autenticación en la petición para acceder a recursos protegidos. Otros

encabezados, como Strict-Transport-Security, indican que el cliente solo debe comunicarse con el servidor utilizando HTTPS.

4. Control de Flujo y Gestión de Sesiones:
    - Encabezados como Cookie y Set-Cookie se utilizan para gestionar las sesiones del usuario y mantener el estado entre múltiples solicitudes, algo que es fundamental en aplicaciones web.
5. Soporte para Diferentes Idiomas y Formatos:
    - Los encabezados como Accept y Accept-Language permiten al cliente indicar al servidor los formatos y lenguajes preferidos para la respuesta, facilitando la personalización de la experiencia del usuario.
6. Optimización y Eficiencia:
    - Los encabezados como Content-Encoding y Transfer-Encoding ayudan a optimizar el tamaño de los datos transmitidos mediante técnicas de compresión como gzip o chunked, reduciendo el tiempo de carga y el consumo de ancho de banda.

5. Cuál es la clasificación de los códigos de estado en la respuesta HTTP

Los códigos de estado HTTP (HTTP Status Codes) son respuestas estándar emitidas por los servidores web en respuesta a las solicitudes realizadas por un cliente (como un navegador o una aplicación).

1. 1xx - Respuestas Informativas (Informational Responses)

Estos códigos indican que la solicitud ha sido recibida y está siendo procesada, pero todavía no se ha completado.

- 100 Continue: El servidor ha recibido la primera parte de la solicitud y el cliente puede continuar enviando el resto.
- 101 Switching Protocols: El servidor acepta cambiar a un protocolo diferente solicitado por el cliente (por ejemplo, cambiar de HTTP a WebSocket).
- 102 Processing: El servidor ha recibido la solicitud y está procesándola, pero aún no hay respuesta disponible (utilizado en WebDAV).
- 103 Early Hints: Sugerencias tempranas sobre los encabezados de respuesta que el servidor tiene la intención de enviar (útil para precargar recursos).

2. 2xx - Éxito (Success)

Estos códigos indican que la solicitud fue recibida, entendida y aceptada correctamente.

- 200 OK: La solicitud ha tenido éxito y el servidor devuelve el recurso solicitado.
- 201 Created: La solicitud ha tenido éxito y ha creado un nuevo recurso. Generalmente, se utiliza en respuestas a peticiones POST.
- 202 Accepted: La solicitud ha sido aceptada para su procesamiento, pero no se ha completado. No se garantiza que se vaya a completar.
- 203 Non-Authoritative Information: La solicitud ha tenido éxito, pero la respuesta ha sido modificada por una fuente distinta al servidor original.
- 204 No Content: La solicitud ha sido exitosa, pero no hay contenido que devolver en la respuesta. Utilizado cuando no se necesita actualizar la página actual.
- 205 Reset Content: Indica al cliente que restablezca el formulario de entrada (similar a 204, pero especifica que el cliente debe reiniciar el estado).
- 206 Partial Content: El servidor devuelve solo una parte del recurso solicitado, generalmente utilizado para la descarga de archivos con rangos (Range header).

3. 3xx - Redirecciones (Redirection)

Estos códigos indican que el cliente debe realizar acciones adicionales para completar la solicitud, como seguir una redirección.

- 300 Multiple Choices: Existen varias opciones para el recurso solicitado. El cliente puede elegir una opción o el servidor sugerirá una predeterminada.
- 301 Moved Permanently: El recurso solicitado ha sido movido de forma permanente a una nueva URL. Se debe utilizar la nueva URL en el futuro.
- 302 Found: El recurso solicitado se encuentra temporalmente en otra ubicación. Se utiliza para redirecciones temporales.
- 303 See Other: El cliente debe realizar una nueva solicitud a otra URL usando el método GET para obtener el recurso. Utilizado para redirección después de POST.
- 304 Not Modified: El recurso solicitado no ha cambiado desde la última solicitud. El cliente puede usar su copia en caché.
- 305 Use Proxy (Obsoleto): El recurso solicitado solo está disponible a través de un proxy. Ya no se utiliza por razones de seguridad.
- 307 Temporary Redirect: El recurso solicitado se ha movido temporalmente a otra URL. El cliente debe usar el mismo método de solicitud (similar a 302, pero con el método POST preservado).
- 308 Permanent Redirect: Similar a 301, pero garantiza que el método de solicitud (como POST) no cambie.

4. 4xx - Errores del Cliente (Client Errors)

Estos códigos indican que la solicitud del cliente contiene errores o no puede ser procesada por alguna razón.

- 400 Bad Request: La solicitud contiene sintaxis incorrecta o no se puede cumplir. Puede ser debido a datos mal formateados.
- 401 Unauthorized: La solicitud requiere autenticación. El cliente debe autenticarse para obtener la respuesta solicitada.
- 402 Payment Required (Experimental): Se reserva para un futuro uso relacionado con pagos. Actualmente no se utiliza de manera estándar.
- 403 Forbidden: El servidor entiende la solicitud, pero se niega a autorizarla. El cliente no tiene permiso para acceder al recurso.
- 404 Not Found: El recurso solicitado no pudo ser encontrado en el servidor. Es uno de los códigos de error más comunes.
- 405 Method Not Allowed: El método de solicitud (como GET o POST) no está permitido para el recurso solicitado.
- 406 Not Acceptable: El recurso solicitado no puede generar una respuesta aceptable según los encabezados Accept enviados en la solicitud.
- 407 Proxy Authentication Required: Similar a 401, pero requiere autenticación a través de un proxy.
- 408 Request Timeout: El servidor no recibió una solicitud completa del cliente dentro del tiempo permitido.
- 409 Conflict: La solicitud no puede ser completada debido a un conflicto con el estado actual del recurso.
- 410 Gone: El recurso solicitado ya no está disponible y no se proporcionará una nueva dirección.
- 411 Length Required: El servidor requiere un encabezado Content-Length en la solicitud.
- 412 Precondition Failed: Una condición preestablecida en los encabezados de la solicitud no se cumple.
- 413 Payload Too Large: El recurso enviado en la solicitud es demasiado grande para ser procesado.
- 414 URI Too Long: La URI proporcionada en la solicitud es demasiado larga para ser procesada por el servidor.
- 415 Unsupported Media Type: El tipo de medio del cuerpo de la solicitud no es soportado por el servidor.

- 416 Range Not Satisfiable: El encabezado de rango de la solicitud no coincide con ningún contenido del recurso.
- 417 Expectation Failed: El servidor no puede cumplir con los requisitos de los encabezados Expect de la solicitud.
- 418 I'm a teapot (RFC 2324, Experimental): Un código humorístico utilizado en respuesta a la solicitud de infusiones de café. No se utiliza en producción.
- 421 Misdirected Request: La solicitud fue dirigida a un servidor que no es capaz de producir una respuesta.
- 426 Upgrade Required: El cliente debe cambiar a un protocolo diferente, como TLS/1.0.
- 429 Too Many Requests: El cliente ha enviado demasiadas solicitudes en un periodo de tiempo determinado (utilizado para limitar peticiones).

5. 5xx - Errores del Servidor (Server Errors)

Estos códigos indican que el servidor encontró una condición que le impide completar la solicitud.

- 500 Internal Server Error: El servidor encontró un error inesperado que impidió cumplir la solicitud.
- 501 Not Implemented: El servidor no reconoce el método de solicitud o no tiene la capacidad de cumplirlo.
- 502 Bad Gateway: El servidor, actuando como una puerta de enlace o proxy, recibió una respuesta inválida del servidor upstream.
- 503 Service Unavailable: El servidor no está disponible temporalmente (por mantenimiento o sobrecarga).
- 504 Gateway Timeout: El servidor, actuando como una puerta de enlace o proxy, no recibió una respuesta oportuna del servidor upstream.
- 505 HTTP Version Not Supported: El servidor no soporta la versión de HTTP utilizada en la solicitud.
- 511 Network Authentication Required: El cliente debe autenticar para poder acceder a la red (por ejemplo, en portales cautivos).