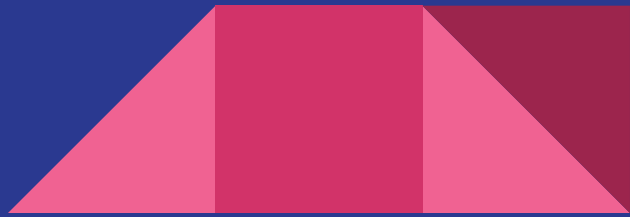
Hypertext Transfer Protocol

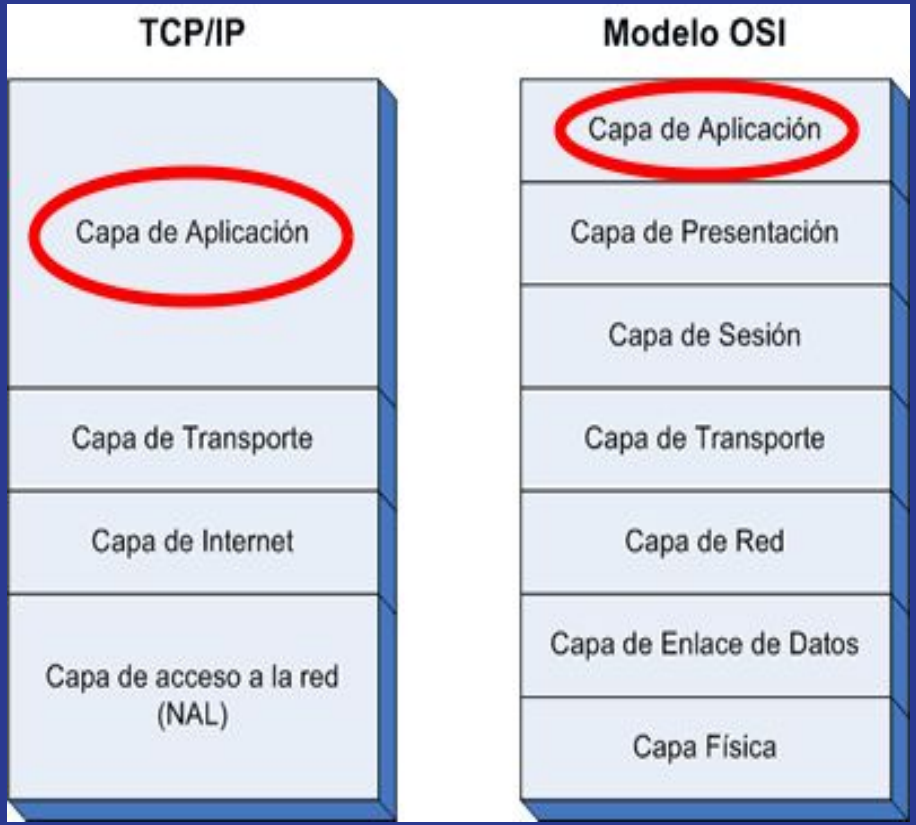
Mario Vega
Jesús Martín

2º DAW

1. Introducción

-HTTP(Hypertext Transfer Protocol) es un protocolo de la capa de aplicación inventado por Tim Berners-Lee en el año 1990, basado en el intercambio de datos(texto, imágenes gráficas, sonido...) mediante una arquitectura de cliente-servidor. Este protocolo se aloja en el puerto 80/TCP.





Este protocolo se sitúa en la capa de aplicación, el séptimo nivel del modelo OSI y el cuarto de la pila TCP.

-¿Cuál es su función?

La principal función del protocolo HTTP es el intercambio de información entre Cliente y Servidor, entre cada petición y respuesta, hay varios intermediarios, normalmente proxies, los cuales realizan distintas funciones, como: gateways o caches, estos intercambios de información se realizan a través de peticiones.

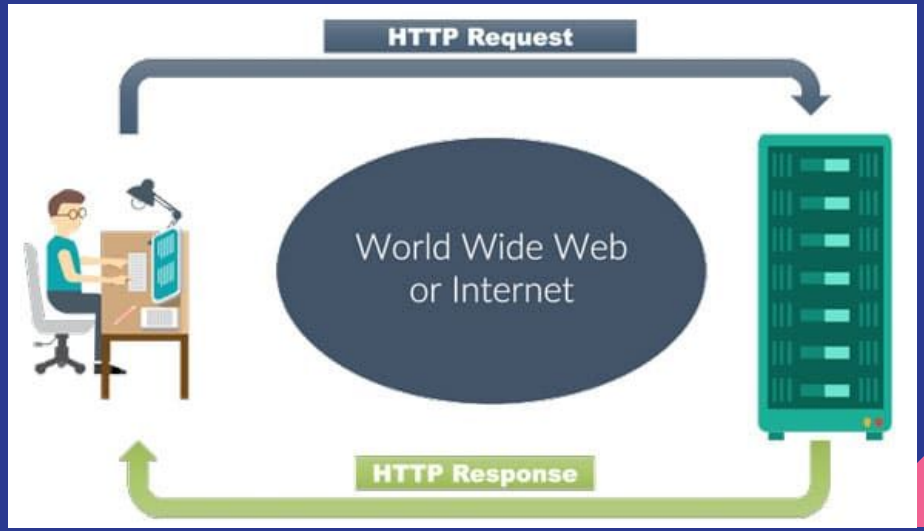
2-.Características

-Sencillez: HTTP está pensado y desarrollado para ser leído y fácilmente interpretado por las personas.

-Extensibilidad: Presentadas en la versión HTTP/1.0, las cabeceras de HTTP, han hecho que este protocolo sea fácil de ampliar y de experimentar con él.

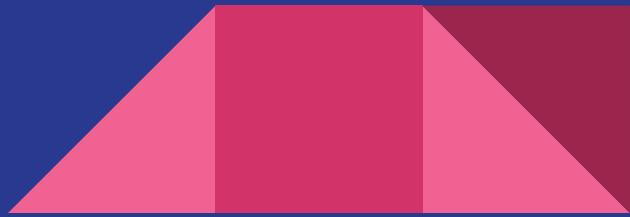
-Protocolo de Sesiones: HTTP es un protocolo sin estado, esto plantea la problemática, en caso de que los usuarios requieran interactuar con determinadas páginas Web de forma ordenada y coherente, pero, aunque HTTP es un protocolo sin estado, el uso de cookies, sí permite guardar datos. Usando la capacidad de ampliación del protocolo HTTP, las cookies permiten crear un contexto común para cada sesión de comunicación.

Es un protocolo sin estado, lo que significa que el servidor no guarda ningún dato entre dos peticiones, es decir, trata cada petición como una transacción independiente que no tiene relación con cualquier solicitud anterior, de modo que la comunicación se compone de pares independientes de solicitud y respuesta.



3-.Métodos de petición

HTTP tiene una serie predefinida de métodos de petición. El protocolo tiene flexibilidad para ir añadiendo nuevos métodos y para así añadir nuevas funcionalidades. Cada método indica la acción que desea que se efectúe sobre el recurso identificado. Lo que este recurso representa depende de la aplicación del servidor.



GET

El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

POST

El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

PUT

El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

DELETE

El método DELETE borra un recurso en específico.

PATCH

El método PATCH es utilizado para aplicar modificaciones parciales a un recurso.

Códigos de respuesta

Los códigos de estado de respuesta HTTP indican si se ha completado satisfactoriamente una solicitud HTTP específica. Las respuestas se agrupan en cinco clases:

- **Códigos con formato 1xx:** Respuestas informativas. Indica que la petición ha sido recibida y se está procesando.
- **Códigos con formato 2xx:** Respuestas correctas. Indica que la petición ha sido procesada correctamente.
- **Códigos con formato 3xx:** Respuestas de redirección. Indica que el cliente necesita realizar más acciones para finalizar la petición.
- **Códigos con formato 4xx**: Errores causados por el cliente. Indica que ha habido un error en el procesado de la petición a causa de que el cliente ha hecho algo mal.
- **Códigos con formato 5xx**: Errores causados por el servidor. Indica que ha habido un error en el procesado de la petición a causa de un fallo en el servidor.

Cabeceras

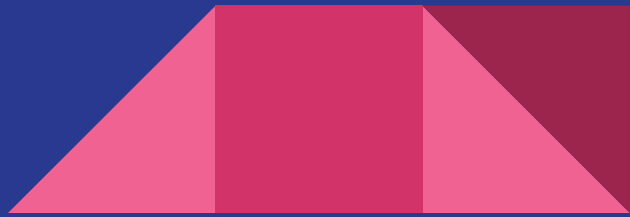
Las cabeceras HTTP permiten al cliente y al servidor enviar información adicional junto a una petición o respuesta.

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.

4-.Generalidades

La característica del protocolo HTTP de ser ampliable, ha permitido que durante su desarrollo se hayan implementado más funciones de control y funcionalidad sobre la Web: caché o métodos de identificación fueron temas que se abordaron pronto en su historia.



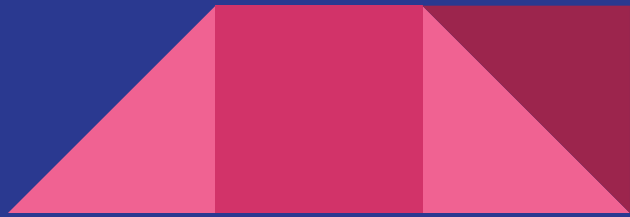
Los elementos que se pueden controlar con el protocolo HTTP:

Flexibilidad del requisito de origen

Para prevenir invasiones de la privacidad de los usuarios, los navegadores Web, solamente permiten a páginas del mismo origen, compartir la información o datos.

Autentificación

Hay páginas Web, que pueden estar protegidas, de manera que solo los usuarios autorizados puedan acceder.

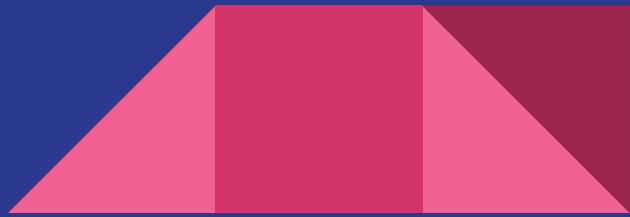


Proxies y tunneling

Servidores y/o clientes pueden estar en intranets y esconder así su verdadera dirección IP a otros.

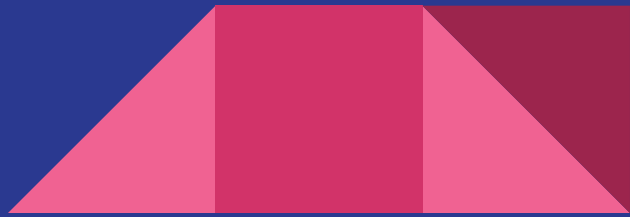
Sesiones

El uso de HTTP cookies permite relacionar peticiones con el estado del servidor. Esto define las sesiones, a pesar de que por definición el protocolo HTTP es un protocolo sin estado.



Caché

El rendimiento de los sitios web y las aplicaciones puede mejorarse significativamente al reutilizar los recursos previamente obtenidos. Los cachés web reducen la latencia y el tráfico de red y, por lo tanto, reducen el tiempo necesario para mostrar una representación de un recurso.

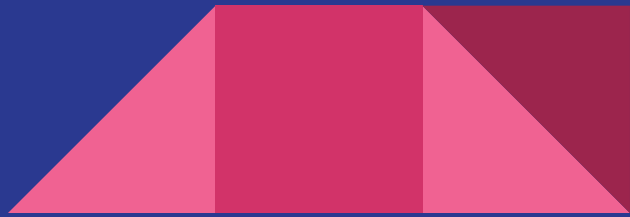


Tipos de caches

Caché privada: contiene todos los documentos descargados a través de HTTP por el usuario.

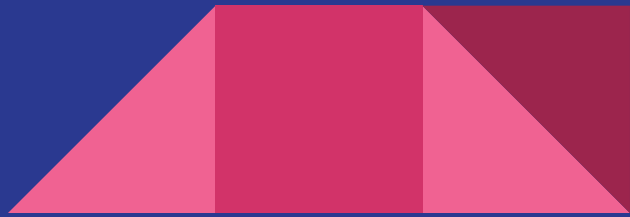
Caché compartida de proxy: Evita viaje adicional al servidor

Caché compartida: es una caché que almacena las respuestas para que sean reutilizado por más de un usuario.



Cookies

Una cookie HTTP, es una pequeña pieza de datos que un servidor envía al navegador web del usuario. El navegador guarda estos datos y los envía de regreso junto con la nueva petición al mismo servidor. Las cookies se usan generalmente para decirle al servidor que dos peticiones tienen su origen en el mismo navegador web lo que permite, por ejemplo, mantener la sesión de un usuario abierta.



Las cookies se utilizan principalmente con tres propósitos:

Gestión de Sesiones

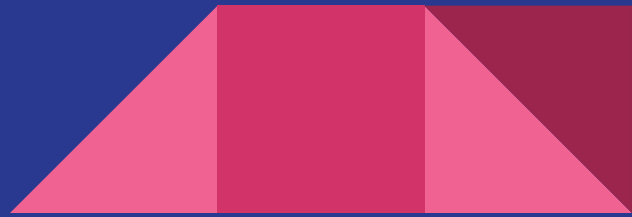
Inicios de sesión, carritos de compras, puntajes de juegos o cualquier otra cosa que el servidor deba recordar

Personalización

Preferencias de usuario, temas y otras configuraciones

Rastreo

Guardar y analizar el comportamiento del usuario



5-. MENSAJES

Petición

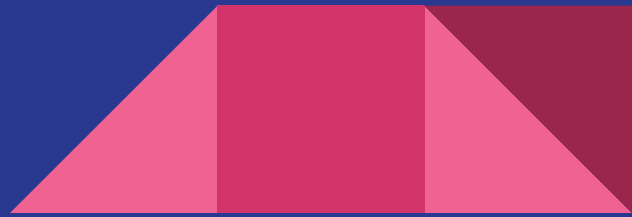
Una petición de HTTP, está formado por los siguientes campos:

-Un método HTTP, que defina la operación que el cliente quiera realizar.

-La dirección del recurso pedido.

-La versión del protocolo HTTP.

-Cabeceras/Cuerpos HTTP (opcionales)



RESPUESTA

Una respuesta HTTP está formada por:

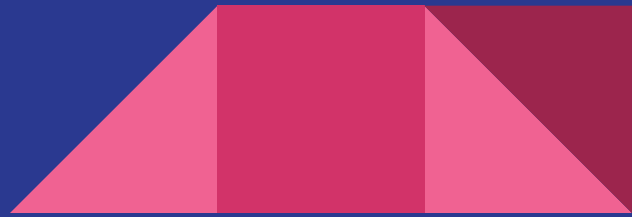
-La versión del protocolo HTTP que están usando.

-Un código de estado, indicando si la petición ha sido exitosa, o no, y debido a qué.

-Un mensaje de estado, una breve descripción del código de estado.

-Cabeceras HTTP, como las de las peticiones.

-Opcionalmente, el recurso que se ha pedido.

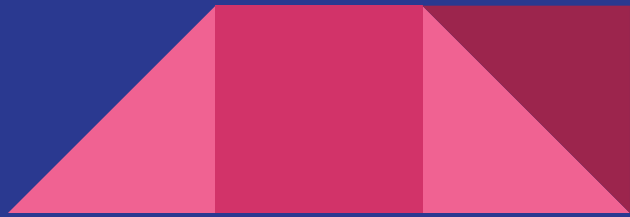


5.1-.PETICIÓN/RESPUESTA

Flujo de datos

Cuando el cliente quiere comunicarse con el servidor, tanto si es directamente con él, o a través de un proxy intermedio, realiza los siguientes pasos:

1. Se abre una conexión en el puerto 80 del host www.example.com .El puerto 80 es el puerto predefinido para HTTP. Si se quisiera utilizar el puerto XXXX habría que añadirlo en la URL de la forma http://www.example.com:XXXX/index.html



2. Se envía un mensaje en el estilo siguiente:

MÉTODO RUTA VERSIÓN HTTP CABECERA

```
GET /index.html HTTP/1.1
Host: www.example.com
Referer: www.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Connection: keep-alive
[Línea en blanco]
```

3. La respuesta del servidor está formada por encabezados seguidos del recurso solicitado, en el caso de una página web:

VERSIÓN

CÓDIGO DE ESTADO Y MENSAJE

CABECERA

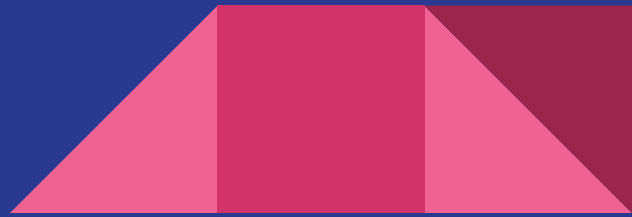
```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html lang="eo">
<head>
<meta charset="utf-8">
<title>Título del sitio</title>
</head>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
   .
   .
   .
</body>
```

6-.SEGURIDAD Y HTTPS

HTTPS es un protocolo de transferencia de datos al igual que HTTP no obstante este es una versión más segura ya que implementa un cifrado de seguridad SSL.

Este protocolo se aloja en el puerto 443 y suele ser usado para la transferencia de datos sensibles



GUÍA DE INSTALACIÓN:

Para realizar la instalación de un servidor HTTP necesitamos dos cosas básicas:

- Tener instalado el apache2 ---> sudo apt install apache2

- Tener una IP estática o un DNS ya que con este podremos acceder a nuestro servidor desde un cliente en nuestra red

GUÍA DE INSTALACIÓN:

Una vez consigamos los requisitos anteriores podremos empezar con la instalación para ello debemos acceder al super usuario y crear una carpeta en el directorio /var/www ya que este es el directorio por defecto de acceso a un servidor HTTP.

```
root@daw2-09:/etc/bind# mkdir -p /var/www/ejemplo.com/html
```

En este caso creamos la carpeta ejemplo.com(nombre de nuestro DNS) y dentro la carpeta HTML en la que guardaremos nuestro fichero HTML el cual será el que veamos desde nuestro cliente

GUÍA DE INSTALACIÓN:

Una vez creada la carpeta pasaremos a editar con nano nuestro fichero HTML como es el caso

```
root@daw2-09:/etc/bind# nano /var/www/ejemplo.com/html/index.html_
```

```
<html>
        <body>
                <h1>Enhorabuena el servidor HTTP funciona correctamente</h1>
        </body>
</html>
```

Es aconsejable llamarlo index.html ya que el servidor busca en determinado orden los ficheros a mostrar en el documento situado en /ETC/APACHE2/MODS-AVAILABLE/DIR.CONF

GUÍA DE INSTALACIÓN:

Una vez configurado nuestro index.html pasaremos a configurar el VirtualHost el cual nos permite asignar a nuestro servidor el puerto de entrada

```
root@daw2-09:/etc# nano /etc/apache/sites-available/ejemplo.com.conf
```

```
<VirtualHost *:80>
        ServerAdmin admin@ejemplo.com
        ServerName ejemplo.com
        ServerAlias www.ejemplo.com
        DocumentRoot /var/www/ejemplo.com/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog $APACHE_LOG_DIR}/access.log combined
</VirtualHost>_
```

GUÍA DE INSTALACIÓN:

Una vez creado el fichero de configuración deberemos habilitar el VirtualHost y deshabilitar el fichero de configuración por defecto para ello utilizaremos los siguientes comandos :

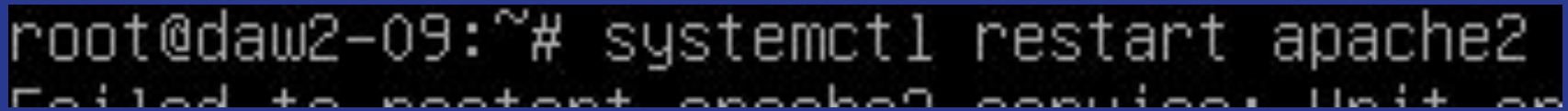
```
root@daw2-09:~# a2ensite ejemplo.com.conf
```

```
root@daw2-09:~# a2dissite 000-default.conf
```

Una vez hecho esto pasaremos a validar los pasos anteriores con el siguiente comando que nos devolverá un SYNTAX OK en el caso de que se hayan realizado correctamente los pasos anteriores.

GUÍA DE INSTALACIÓN:

Ya por último pasaremos a reiniciar el apache2 para que nuestro servidor funcione con el siguiente comando:



Una vez completados estos pasos podremos comprobar la instalación accediendo desde nuestro cliente a la IP de nuestro servidor o a su DNS y este nos mostrará en el navegador nuestro index.html.