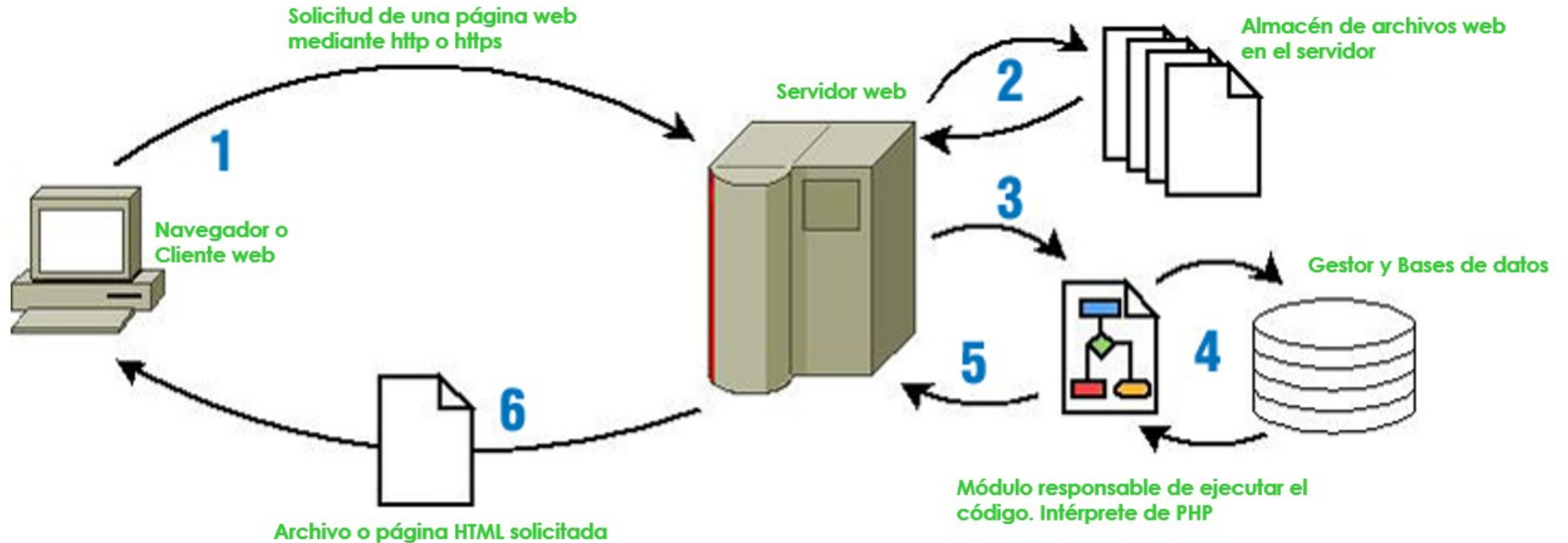




# **Encabezados HTTP**

## El esquema de funcionamiento de una página web dinámica en el entorno del servidor



# Encabezados: palabras en una conversación

HTTP es el protocolo (el conjunto de 'reglas') para transferir datos entre los servidores web y los navegadores de clientes, y generalmente se utiliza el puerto 80 para realizar esta comunicación.

El cliente web (navegador) localiza al servidor web mediante una cadena de caracteres denominada **dirección URL**.

Aquí es donde aparece el "**http://**" en las URL de los sitios web.

El propósito del protocolo HTTP **es permitir la transferencia de archivos entre un navegador (el cliente) y un servidor web.**

# Encabezados: palabras en una conversación

- Los encabezados HTTP son una parte fundamental de la comunicación en la web, y juegan una función crucial en cómo se transfiere información entre el navegador (cliente) y el servidor web.
- Aunque son invisibles para el usuario, los encabezados llevan instrucciones esenciales para controlar la forma en que se entregan los recursos, como páginas web, imágenes, archivos de video, ...

## ¿Qué son los encabezados HTTP?


- Un encabezado HTTP **es un pequeño fragmento de información** que **acompaña cada solicitud** que el navegador envía al servidor y cada respuesta que el servidor devuelve al navegador.
- Estos encabezados **contienen metadatos sobre la solicitud o respuesta**, como el tipo de contenido que se envía, la codificación, el idioma, las políticas de caché, la autenticación, ...

# Encabezados: palabras en una conversación

## Comunicación entre cliente y servidor

Cuando un usuario accede a una página web, el proceso sigue estos pasos fundamentales:

1. El cliente (navegador) realiza una solicitud HTTP al servidor web. Este proceso comienza cuando un usuario escribe una URL en la barra de direcciones o hace clic en un enlace.
2. El servidor recibe esa solicitud, la procesa y envía una respuesta HTTP al cliente. Esta respuesta contiene no solo los datos solicitados (por ejemplo, una página HTML), sino también una serie de encabezados HTTP que describen cómo debe manejarse esa información.



HTTP es un  
protocolo  
basado en  
mensajes  
texto

Los encabezados se pueden dividir en dos tipos:

- Los encabezados de **Petición (Request)** que el navegador envía al servidor cuando solicita un archivo
- Los encabezados de **Respuesta (Response)** que el servidor envía al navegador cuando sirve el archivo.




# Encabezados de petición

## *Request headers*

### Encabezados de solicitud (Request Headers):

- **User-Agent:** Este encabezado proporciona información sobre el navegador y el sistema operativo que el usuario está utilizando. Los servidores web pueden utilizar esta información para adaptar la respuesta a las capacidades del navegador y garantizar una experiencia óptima.
- **Referer (Referencia):** Indica la página desde la cual se realizó la solicitud. Esto es útil para los sitios web que desean saber cómo los usuarios llegaron a una página en particular, como a través de un enlace o una búsqueda. *Error tipográfico que se debe mantener. Lo correcto sería **referrer**.*
- **Cookies:** Las cookies son pequeños fragmentos de información que se almacenan en el navegador y se envían al servidor con cada solicitud. Estas cookies pueden contener información de autenticación, preferencias del usuario y datos de seguimiento.
- **Accept-Language (Aceptar-Lenguaje):** Este encabezado indica los idiomas que el navegador del usuario está dispuesto a aceptar, lo que permite a los servidores web ofrecer contenido en el idioma preferido del usuario



# Encabezados de respuesta

## *Response headers*

### Encabezados de respuesta (Response Headers)

- **Content-Type (Tipo de contenido):** Este encabezado especifica el tipo de contenido que se envía al navegador, como HTML, imágenes, archivos CSS o JavaScript. El navegador utiliza esta información para interpretar y representar correctamente el contenido.
- **Content-Length (Longitud del contenido):** Indica la longitud en bytes del contenido que se envía al navegador. Esto ayuda al navegador a gestionar la descarga de recursos.
- **Cache-Control:** Controla cómo se almacena en caché el contenido en el navegador o en servidores proxy intermedios. Esto puede acelerar la carga de páginas web al permitir que el navegador almacene en caché recursos para su uso posterior.
- **Location (Ubicación):** En caso de redirecciones, este encabezado indica la nueva ubicación a la que debe dirigirse el navegador.
- **Set-Cookie (Establecer cookie):** Si el servidor desea almacenar una cookie en el navegador del usuario, utiliza este encabezado para enviar la información de la cookie.



# Ejemplo del ciclo de solicitud y respuesta HTTP

## Solicitud HTTP

El navegador (cliente) envía una solicitud con encabezados HTTP, como:

- User-Agent: Describe el navegador y su versión.
- Accept: Indica los tipos de contenido que el navegador puede procesar (por ejemplo, HTML, JSON, XML).
- Host: Especifica el dominio del servidor al que se está haciendo la solicitud.

## Ejemplo de una solicitud HTTP:

```
GET /index.html HTTP/1.1  
Host: www.ejemplo.com  
User-Agent: Mozilla/5.0  
Accept: text/html
```

# Ejemplo del ciclo de solicitud y respuesta HTTP

## Respuesta HTTP


El servidor responde con encabezados HTTP, como:

- Content-Type: Indica el tipo de archivo que se envía (por ejemplo, text/html para una página web, image/jpeg para una imagen).
- Content-Length: Especifica el tamaño del archivo en bytes.
- Cache-Control: Controla cómo y cuándo el contenido debe ser almacenado en caché.

### Ejemplo de una respuesta HTTP:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1024
Cache-Control: no-cache
```

- El encabezado `HTTP/1.1 200 OK` es una **línea de estado en la respuesta HTTP** enviada por un servidor web al cliente (el navegador) que indica que la solicitud se ha procesado con éxito.



Los encabezados se envían antes del contenido principal y están separados del contenido principal por una línea en blanco

El nombre del encabezado va primero, seguido de dos puntos y, a continuación, contenido o valor del encabezado:

**Header-Name:** header-value

**Location:** <http://www.miservidor/documento.html>

# El formato de los encabezados headers

- **Línea inicial del mensaje**
  - Primera línea del mensaje donde se indica qué hacer (mensaje de petición) o qué ha ocurrido (mensaje de respuesta).
  - La primera del mensaje se conoce como "**línea de solicitud**" en una petición o "**línea de estado**" en una respuesta.
- **Cabecera del mensaje**
  - Contienen los atributos del mensaje.
  - Bloque de campos terminados por **una línea en blanco**
- **Línea en blanco**
- **Cuerpo del mensaje**
  - El cuerpo del mensaje es una **sección opcional** que sigue a la cabecera.
  - Su presencia y contenido dependen del tipo de recurso solicitado o del resultado de la solicitud.
  - **En una petición**, el cuerpo puede contener datos que se envían al servidor, como el contenido de un formulario.
  - **En una respuesta**, el cuerpo contiene el recurso solicitado, que puede ser una página HTML, una imagen, un archivo de texto, etc.

# El formato de los encabezados headers Línea en blanco

- La línea en **blanco** es una línea vacía, sin caracteres visibles, que sigue a la cabecera del mensaje.
- Su principal función **es separar la cabecera del cuerpo** del mensaje, lo que permite al receptor del mensaje, ya sea un navegador o un servidor, saber dónde termina la información de la cabecera y comienza el contenido real del recurso.
- La línea en blanco es esencial para el procesamiento adecuado de la solicitud o respuesta, ya que establece una clara distinción entre los metadatos y los datos.

**En una petición,** la línea inicial contiene información sobre la acción que se debe realizar y el recurso objetivo. La estructura es la siguiente:

```
<Método> <URL> <Versión de HTTP>
```

```
GET /pagina.html HTTP/1.1
```

**En una respuesta,** la línea inicial indica el resultado del procesamiento de la solicitud y la versión de HTTP utilizada. La estructura es similar:

```
<Versión de HTTP> <Código de estado> <Frase de estado>
```

```
HTTP/1.1 200 OK
```

**El bloque de cabecera** sigue inmediatamente a la línea inicial y contiene una serie de campos con sus respectivos valores que proporcionan atributos y metadatos del mensaje.  
Cada campo tiene un formato de **Nombre: Valor**

- **Ejemplo de cabecera de solicitud**

```
Host: www.ejemplo.com  
User-Agent: Mozilla/5.0  
Accept-Language: en-US
```

- **Ejemplo de cabecera de respuesta**

```
Content-Type: text/html  
Content-Length: 2048  
Server: Apache/2.4
```

La cabecera del mensaje es un bloque de campos que finaliza con **una línea en blanco**.



**El bloque cuerpo del mensaje:** El cuerpo del mensaje es una sección opcional que sigue a la cabecera. Su presencia y contenido dependen del tipo de recurso solicitado o del resultado de la solicitud.

Ejemplo de **cuerpo del mensaje de solicitud** en el caso de que se envíe un formulario

```
nombre=Juan&email=juan@example.com
```

- El formato más común para enviar datos de formulario es la codificación **application/x-www-form-urlencoded**.
- En este formato, los datos del formulario se envían como pares clave-valor separados por el símbolo "&".
- **nombre** es el nombre del primer campo del formulario, y su valor es "**Juan**".
- **email** es el nombre del segundo campo del formulario, y su valor es "**juan@example.com**".
- Los pares **clave-valor** están separados por el símbolo **&**, lo que permite al servidor entender que se trata de dos campos diferentes con sus respectivos valores.

**Cuerpo del mensaje de respuesta:** Contiene el recurso real solicitado. Por ejemplo:

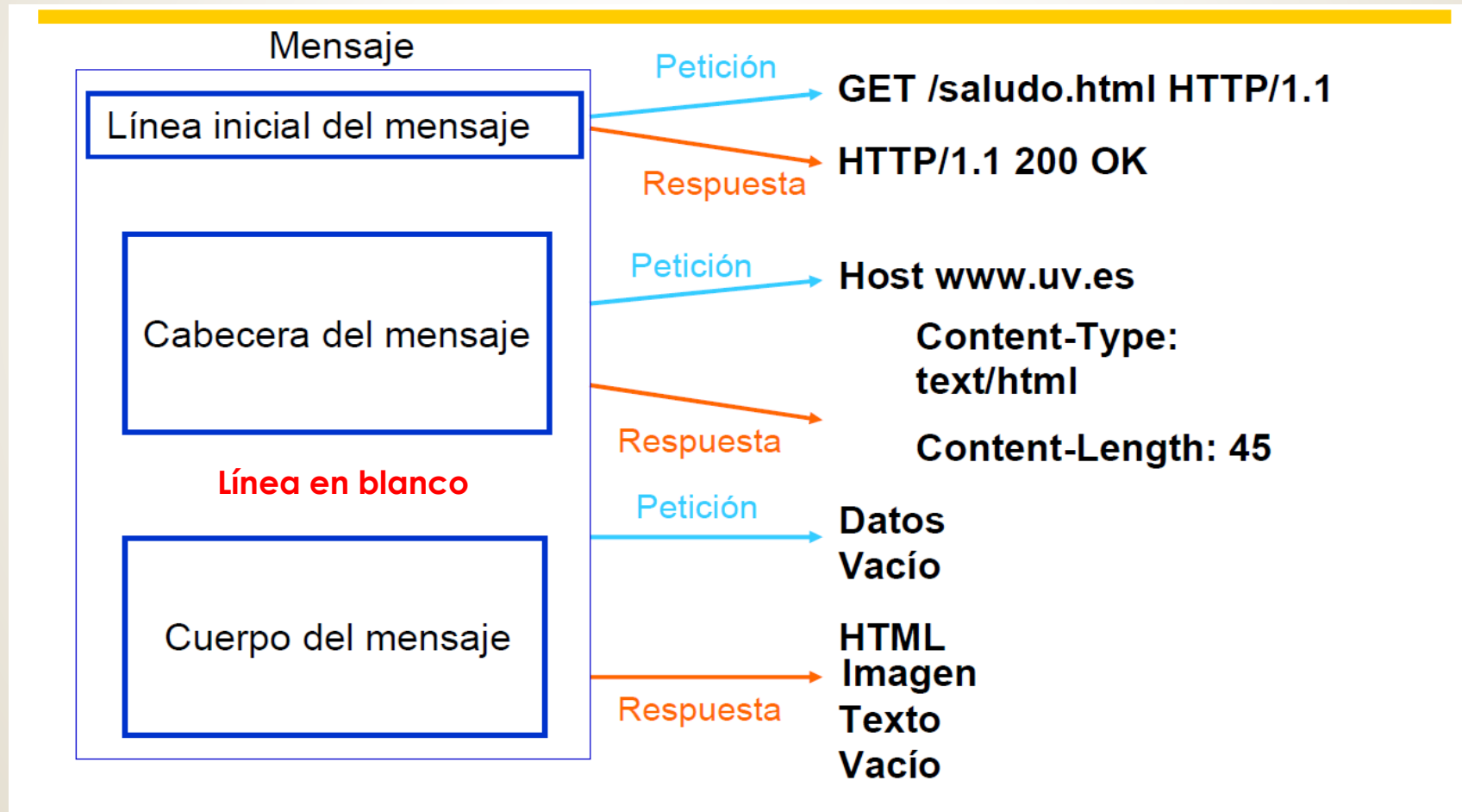
- El contenido de una página web
- Una imagen
- Un archivo de texto...



## Estructura de los encabezados de solicitud y respuesta

Ejemplo cuando un navegador solicita la siguiente URL:

<http://www.uv.es/saludo.html>



# Estructura de un mensaje HTTP

---

## Línea Inicial del Mensaje

Ejemplo:

```
GET /pagina.html HTTP/1.1
```

---

## Cabecera del Mensaje

Ejemplo:

```
Content-Type: text/html  
Content-Length: 1234  
Connection: keep-alive
```

---

## Línea en Blanco

---

## Cuerpo del Mensaje

(Opcional)

Ejemplo:

```
<html>  
  
<head>  
  <title>Página de Ejemplo</title>  
</head>  
  
<body>  
  <h1>Hola Mundo!</h1>  
</body>  
  
</html>
```

# PHP y la Generación Automática de Encabezados HTTP

PHP fue diseñado desde sus inicios para facilitar la generación de HTML, lo que incluye la gestión de encabezados HTTP. Cuando un script PHP se ejecuta, se encarga de enviar automáticamente los encabezados HTTP correspondientes al cliente (navegador) en el momento en que se genera la primera salida del script.

## 1. Generación Automática de Encabezados:

- La primera vez que un script PHP produce una salida (por ejemplo, mediante la función echo o print):
  - PHP genera automáticamente los encabezados HTTP necesarios para la respuesta.
  - Esto incluye encabezados como Content-Type, que indica el tipo de contenido que se está enviando al cliente (por defecto, **text/html** para una página web).

## Ejemplo:

- Script:

```
echo "<h1>Hola Mundo!</h1>";
```

**PHP generará automáticamente los encabezados HTTP en ese momento**, sin que el programador tenga que hacerlo manualmente. Esto simplifica el proceso de desarrollo, pero tiene sus inconvenientes.

## 3. Confusión Común:

- Esta automatización es una gran ayuda, pero también puede llevar a confusiones. Muchos programadores novatos en PHP no comprenden completamente cómo y cuándo se envían los encabezados.
- Por ejemplo, si intentan enviar encabezados manualmente después de haber producido una salida, recibirán un error porque los encabezados HTTP solo se pueden enviar antes de cualquier salida.

Función `header()`  
función de php  
que envía  
encabezados de  
respuesta HTTP

# header()

Con esta función, podemos hacer que los scripts envíen encabezados de nuestra elección al navegador y crear algunos resultados deseados.

IMPORTANTE: la función header() **hay que usarla antes de que PHP envíe cualquier salida** (antes de que se envíen los encabezados predeterminados de PHP).

Dudo que haya un programador de PHP en el mundo que nunca haya visto un error como este:

Warning: Cannot modify header information - headers already sent by...

*Advertencia: no se puede modificar la información del encabezado - los encabezados ya enviados por...*

Los encabezados de respuesta están separados del contenido por una línea en blanco y esto significa **que solo puede enviarlos una vez**. Después de esa línea en blanco va el contenido, **NO** puede haber más encabezados.

Si el script tiene algún resultado, incluso una línea o espacio en blanco antes de: `<?php`

**PHP envía el resultado sin preguntar y junto a ese resultado genera los encabezados predeterminados.**

El siguiente script:

```
<?php
// Enviar suficiente contenido para llenar el buffer
for ($i = 0; $i < 1000; $i++) {
    echo "Llenando el buffer de salida... "; // Enviar texto repetido
}
$test = NULL;
if ($test) {
    echo "Estás dentro!";
} else {
    header('Location: https://www.iesvenancioblanco.es/');
}
```

- Lo que esta secuencia de comandos está tratando de hacer es redirigir al usuario utilizando el encabezado location si **\$test** no es cierto. Una vez mostrados los mensajes.
- ¿Cuál es el problema? El texto “*Llenando el buffer de salida...*” se envía primero, por lo tanto, los encabezados predeterminados se envían automáticamente.
- En el momento en que llamamos a la función **header()**, ya es demasiado tarde: en lugar de ser redirigido, el usuario verá un mensaje de error.

## Directiva **output\_buffering** de Apache

El buffer de salida del intérprete PHP se crea mediante la directiva **output\_buffering**.

El buffer de salida almacena temporalmente la salida del programa, y no se envía al navegador hasta que se complete la página (finalice el script) o se llene el buffer.

Si la salida del programa no se envía al navegador, tampoco se envían los encabezados predeterminados en ese momento.

A través de la directiva **output\_buffering** del fichero “**php.ini**” es posible activar un buffer de salida.

**Normalmente está activada**, como en nuestro caso, pero tenéis que recordar que cuando publicamos una página en un servidor privado dependemos de como esté configurado.

Líneas extraídas de “**php.ini**”


```
; Development Value: 4096
; Production Value: 4096
; http://php.net/output-buffering
output_buffering = 4096
```

También se puede comprobar su estado a través de la función `phpinfo()`

open_basedir	no value	no value
output_buffering	4096	4096
output_encoding	no value	no value
output_handler	no value	no value


Cuando se hace uso de este buffer de salida, lo que genera el script de php no se envía directamente al cliente(navegador) **sino que se queda en esta memoria intermedia, permanece aquí hasta que el script termina de ejecutarse o se llena el buffer.**





Activar el almacenamiento  
en búfer de la salida para  
un script cuando la  
directiva


**output\_buffering**  
**esté desactivada**



# Activar un búfer de salida en un script


- **El búfer de salida** permite mantener la salida de datos (incluso etiquetas HTML fuera del código PHP) y enviarlo al navegador sólo cuando se le indica explícitamente.
- De esta manera, puedes programar lo que quieras y enviar explícitamente la salida después de haber especificado los encabezados que necesitas.
- Las funciones relevantes son:
  - ob\_start()**, que activa el búfer de salida,
  - ob\_get\_clean()** vuelca el contenido del búfer en una variable.
  - ob\_flush()**: vuelca el contenido del búfer.
  - flush()** envía el contenido del búfer al navegador.
  - ob\_end\_flush()** cierra el búfer.

`flush()` te permite enviar el contenido parcialmente sin cerrar el búfer, y `ob_end_flush()` envía todo el contenido y cierra el búfer.



## Activar un búfer de salida en un script

- **El búfer de salida** en PHP permite que la salida de un script se almacene temporalmente en memoria antes de enviarse al navegador (incluso etiquetas HTML fuera del código PHP) y enviarlo al navegador sólo cuando se le indica explícitamente.
- Esto significa que, en lugar de enviar inmediatamente la salida conforme se genera, PHP puede retenerla en un búfer, lo que te da la capacidad de manipular la salida, organizarla o cambiarla antes de que sea enviada al cliente.
- Y de esta manera, puedes programar lo que quieras y enviar explícitamente la salida después de haber especificado los encabezados que necesitas.



# Funciones para manejar el búfer en PHP

**ob\_start():** Activa el búfer de salida, toda la salida del script se retiene en el búfer.

**ob\_get\_clean():** Vuelca el contenido del búfer en una variable y lo limpia, pero no envía nada al navegador.

**ob\_flush():** Envía el contenido del búfer al navegador, pero **no** la salida de echo o print, pero sigue manteniendo el búfer abierto.

**flush():** Asegura que todo el contenido del búfer (incluso de bajo nivel: echo y print) se envíe al navegador. Útil junto con **ob\_flush()**.

**ob\_end\_flush():** Envía el contenido del búfer al navegador y lo cierra. El búfer deja de estar activo

# Funciones para manejar el búfer en PHP

**ob\_start():** Activa el búfer de salida, toda la salida del script se retiene en el búfer.

**ob\_get\_clean():** Vuelca el contenido del búfer en una variable y lo limpia, pero no envía nada al navegador.

**ob\_flush():** Envía el contenido del búfer al navegador, pero **no** la salida de echo o print, pero sigue manteniendo el búfer abierto.

**flush():** Asegura que todo el contenido del búfer (incluso de bajo nivel: echo y print) se envíe al navegador. Útil junto con **ob\_flush()**.

**ob\_end\_flush():** Envía el contenido del búfer al navegador y lo cierra. El búfer deja de estar activo

Se debe vaciar el búfer de salida tan pronto como sea posible, especialmente si tiene bastante contenido para enviar.

De lo contrario, **la página parecerá cargarse más lentamente**, ya que el contenido se enviará sólo después de que se haya llenado el búfer o finalizado el script, en lugar de estar disponible desde la ejecución del código.

```
<?php
// Mostrar texto directamente en el navegador
echo "Este texto se muestra directamente en el navegador, no he activado el buffer.<br>";
echo "Ahora activo el buffer <code>ob_start()</code><hr>";
// Iniciar el almacenamiento en búfer de salida
ob_start();
// Generar contenido que se almacena en el búfer
?>
Bienvenido a mi sitio!
<?php
echo "<br>Este texto se almacena en el búfer.";
// Más salida generada, se almacena en el búfer
echo '<br>Este es otro texto en el búfer y que con <code>ob_get_clean()</code> limpio el buffer y guardo su contenido en la variable $content.';
echo '<br>Si quiero mostrar las frases desde que activé el buffer debo hacer <code>echo</code> de la variable <code>$content</code><br><hr>';

// Recuperar el contenido del búfer y para enviarlo al navegador
$content = ob_get_clean();
echo $content;

// Mostrar el contenido del búfer
echo "Esta frase está después de limpiar el buffer y la mostrará <code>flush()</code><hr>";
ob_flush(); // Vaciar el búfer de salida
flush(); // Enviar el contenido parcial del búfer al navegador

// Pausar la ejecución del script durante 5 segundos
sleep(5);
// Generar más contenido después de vaciar el buffer y dormir 5 segundos el script
echo "Este texto se muestra al cerrar el buffer y pasados 5 segundos porque hay un <code>sleep(5)</code> ";
echo "<br>Si hay algún contenido almacenado en el búfer de salida, <code>ob_end_flush()</code> lo envía al navegador y cierra el buffer.";
// Cerrar el búfer
ob_end_flush();
?>
```

# Encabezado HTTP Location



# Redirigiendo con el encabezado: **Location**

- Redirecciona a una página indicada

```
<?php
// Redirigir a una página diferente
header('Location: http://www.miservidor/documento.html');
exit; // Asegurarse de que el script se detenga después de la redirección
?>
```

- De acuerdo con la especificación HTTP 1.1, realmente se debería usar una **URL absoluta** con el encabezado Location, si bien es posible que funcione con una **URL relativa** para el valor.
- Para que forme correctamente una ruta absoluta debe incluir **http://** como parte de la ruta



- Un error que es fácil cometer con el encabezado Location es no llamar **exit** directamente después.
- Es posible que no siempre quiera hacer esto, pero generalmente sí.
- **La razón por la que esto es un error es que el código PHP de la página continúa ejecutándose, aunque el usuario haya ido a una nueva ubicación y no vea el resultado del resto de la ejecución.**
- En el mejor de los casos, esto utiliza recursos del sistema innecesariamente. En el peor de los casos, puedes realizar tareas que nunca quisiste hacer. Considera el siguiente código:

```
<?php
//Redirigir a otra página a los usuarios sin permiso
if ( $username < 4){
    header('Location: http://www.misito.com/otrapagina.php');
}

//Enviar un mail con el código secreto
enviar_codigo_secreto($username); //supongamos una función que envía al correo
echo 'El código secreto ha sido enviado';
?>
```

# Encabezado HTTP Refresh

## Redirigiendo con el encabezado Refresh

Los Refresh redireccionan como lo hace el encabezado Location, pero puede agregar un retraso antes de que el usuario sea redirigido.

Por ejemplo, el siguiente código redirigiría al usuario a una nueva página después de mostrar la actual durante 10 segundos:

```
<?php
header('Refresh: 10; url=https://www.iesvenancioblanco.es/index.php');
echo 'Serás redirigido en 10 segundos';
exit();
?>
```

## Redirigiendo con el encabezado Refresh

Otra aplicación común es forzar una página para que se actualice repetidamente "redirigiendo" a la página actual.

Por ejemplo, una página que contará desde 10, con una pausa de 2 segundos entre los números:

```
<?php
// Verificar si el parámetro 'n' no está establecido en la URL
if (!isset($_GET['n'])) {
    // Si no está establecido, inicializar 'n' a 10
    $n = 10;
    $_GET['n'] = 10;
} else {
    // Si está establecido, asignar el valor de 'n' desde la URL
    $n = $_GET['n'];
}

// Verificar si 'n' es mayor que 0
if ($n > 0) {
    // Enviar un encabezado HTTP para refrescar la página cada 3 segundos
    // y decrementar el valor de 'n' en 1
    header('Refresh:2;url=' . $_SERVER['PHP_SELF'] . '?n=' . ($n - 1));

    // Mostrar el valor actual del contador
    echo "<br><center><h1>Valor del contador: " . $_GET['n'] . "</h1></center>";
} else {
    // Si 'n' es 0 o menor, mostrar el mensaje de fin del contador
    echo "<br><center><h1>FIN del contador</h1></center>";
}
```

# Encabezados HTTP

## Content-type

## Crear con PHP otros tipos de archivos

- El servidor asigna **por defecto el tipo MIME text/html** a los ficheros generados por un programa PHP.
- PHP puede generar archivos con cualquier contenido (archivos de texto o incluso binarios), pero para que el navegador acepte esos archivos y los interprete correctamente, en las cabeceras se debe incluir el tipo MIME correspondiente.
- El tipo MIME se indica en la cabecera, por lo que un programa PHP puede declarar el tipo MIME mediante la función `header()`, enviando como argumento la cadena **Content-type:** seguida del tipo MIME correspondiente.
- Url a tabla con los tipos MIME:

[https://developer.mozilla.org/es/docs/Web/HTTP/MIME\\_types/Common\\_types](https://developer.mozilla.org/es/docs/Web/HTTP/MIME_types/Common_types)



## Encabezado **Content-Type**

El encabezado **Content-Type** es esencial en la comunicación entre servidores web y navegadores, ya que indica al navegador qué tipo de datos está recibiendo, lo que permite que el navegador los interprete correctamente.

Dependiendo del MIME type especificado, el navegador puede decidir cómo manejar el contenido:

- Mostrarlo directamente en el navegador
- Descargarlo
- Abrirlo con una aplicación externa o una extensión de navegador.

# Encabezado

## Content-Type

### Tipos MIME

***Multipurpose Internet Mail Extensions*** o **MIME** ("extensiones multipropósito de correo de internet") es una forma estandarizada de indicar la naturaleza y el formato de un documento, archivo o conjunto de datos.

La estructura de un tipo MIME es muy simple; consiste en un tipo y un subtipo, dos cadenas, separadas por un /. No se permite espacio. NO distingue entre mayúsculas y minúsculas, pero tradicionalmente se escribe todo en minúsculas.

Ejemplos comunes de tipos MIME:

➤ Páginas HTML o PHP:

- Tipo MIME: text/html

```
Content-Type: text/html; charset=UTF-8
```

➤ Imágenes:GIF:

- Tipo MIME: image/gif

```
Content-Type: image/gif
```

➤ Archivos PDF:

- Tipo MIME: application/pdf

```
Content-Type: application/pdf
```

➤ Archivos ZIP:

- Tipo MIME: application/zip

```
Content-Type: application/zip
```



Extensión	Tipo de MIME	Tipo de Documento
.html / .htm	text/html	Documento HTML
.xml	text/xml	Documento XML
.css	text/css	Hoja de estilos CSS
.json	application/json	Formato JSON
.pdf	application/pdf	Documento PDF
.zip	application/zip	Archivo ZIP
.jpeg / .jpg	image/jpeg	Archivo de imagen jpeg o jpg
.avi	video/x-msvideo	Archivo de video avi
.bin	application/octet-stream	Archivo binario
.doc	application/msword	Archivo Microsoft Word
.epub	Application/epub+zip	Publicación electrónica(EPUB)
.icon	Image/x-icon	Formato Icon
.js	Application/javascript	Documento Javascript
.mpeg	Video/mpeg	Archivo de video mpeg
.mp4	Video/mp4	Archivo de video mp4
.wav	Audio/x-wav	Archivo de audio wav
.gif	Image/gif	Archivo de imagen gif
.png	Image/png	Archivo de imagen png
.3gp	Video/3gp	Archivo de video 3gpp
.xls	Application/vnd.ms-excel	Archivo Microsoft Excel
.odt	Application/vnd.oasis.open document.text	Archivo de texto OpenDocument



## Encabezado Content-Type

### Uso de Content-Type en PHP

PHP puede generar cualquier tipo de contenido dinámicamente, desde texto plano hasta archivos binarios, estableciendo el encabezado Content-Type adecuado.

Por ejemplo:

Usando este encabezado, se puede hacer que los scripts PHP envíen cualquier tipo de archivo, desde archivos de texto sin formato a imágenes o archivos zip.

## Encabezado Content-Type

Por ejemplo, se puede enviar al usuario un archivo de texto pre-formateado en lugar de HTML:

```
<?php
$variable = "<h1>esto es texto plano</h1>";
header('Content-Type:text/plain');
echo $variable;
?>
```

Utilizad el botón derecho del ratón para ver el código fuente que ha recibido el navegador

# Encabezado HTTP

## Content-Disposition



# Content-disposition

■ El encabezado Content-Disposition indica si el contenido se espera que se muestre el navegador:

■ como una parte de una página web

■ como un archivo adjunto, que se puede descargar y guardar localmente.

# Content-disposition

El primer parámetro puede ser:

- **inline** :valor predeterminado, indicando que puede ser mostrado dentro de una página web

```
<?php
// La función header() envía una cabecera HTTP a un cliente.
// 'Content-Disposition: inline' sugiere al navegador que el archivo debe ser mostrado en el navegador.
header('Content-diposition: inline);

// La cabecera 'Content-type: text/plain' le dice al navegador que el contenido es texto plano.
header('Content-type: text/plain');

// La función readfile() lee un archivo y lo escribe en el buffer de salida.
// readfile() se lee y envía directamente al navegador del cliente como una respuesta HTTP.
// 'archivo.txt' es el nombre del archivo a leer, puede indicarse la ruta
readfile('archivo.txt');
?>
```

# Content-disposition

**El primer parámetro puede ser:**

- **attachment** :indicando que será descargado; la mayoría de los navegadores mostrarán un diálogo 'Guardar como'. Admite prellenado(sugerencia de nombre de archivo) con el valor del parámetro filename, en caso de estar presente.
- **attachment** indica al navegador que abra una caja de descarga de archivos, en lugar de analizar el contenido.

Content-Disposition: attachment

Content-Disposition: attachment; filename="filename.jpg"

# Encabezado Content-Type

```
<?php
//Función para leer el fichero indicado, puede contener ruta
//Guardamos el contenido del fichero en una variable
$contenidoPdf=file_get_contents('papelera.png');

//Indica tipo de fichero que va a recibir el navegador

header("Content-type: image/png");

//Muestra cuadro de diálogo para la descarga
header("Content-Disposition: attachment; filename=NombreSugerido.png");

//Muestra el archivo en el navegador
//header("Content-Disposition: inline; filename=NombreSugerido.pdf");

//Indica el tamaño del fichero que va a recibir el navegador
header('Content-Length: ' . strlen($contenidoPdf));

//Muy importante esta instrucción ya que si no se envía el navegador no sabrá
echo $contenidoPdf;

?>
```



# Cabecera HTTP Content-Length



- Cuando el contenido se va a enviar al navegador, el servidor puede indicar el tamaño en bytes:

```
header( 'Content-Length:123245' );
```

- Esto es especialmente útil para la descarga de archivos, así el navegador puede calcular el progreso de la descarga.


```
<?php
$ruta = 'misEjemplos';
// Nombre del archivo .rar
$rarArchivo = 'archivo.rar';
// Ruta completa al archivo
$rutaCompletaRar = $ruta . '/' . $rarArchivo;

// Verificamos si el archivo existe
if (file_exists($rutaCompletaRar)) {
    // Obtenemos el contenido del archivo y lo guardamos en una variable
    $contenidoFichero = file_get_contents($rutaCompletaRar);

    // Establecemos las cabeceras para indicar que es un archivo .rar
    header('Content-Type: application/x-rar-compressed'); // Tipo MIME para archivos .rar
    header('Content-Disposition: attachment; filename="' . pathinfo($rutaCompletaRar, PATHINFO_BASENAME) . '"');
    header('Content-Length: ' . filesize($rutaCompletaRar));
    // Enviamos el contenido del archivo al navegador. IMPORTANTE: No hacer echo de nada antes de esta línea
    // Si no se hace echo el fichero no se enviará al navegador
    echo $contenidoFichero;
    exit;
} else {
    echo "El archivo no existe.";
}

//Content-Disposition: attachment
//Este encabezado le indica al navegador que el archivo debe ser tratado como un archivo adjunto y no como un archivo que
se debe mostrar directamente en el navegador.
//Al usar attachment, el navegador forzará la descarga del archivo, en lugar de intentar abrirlo dentro de la ventana del
navegador

//filename
// Este parámetro define el nombre que el navegador debe sugerir para guardar el archivo cuando el usuario lo descarga.
//Al incluir filename="nombre.extensión", le indicas al navegador cómo debe llamar al archivo descargado.
```



Cabeceras para  
gestionar el  
almacenamiento  
en caché

# Encabezado HTTP

## Cache-Control



# Caché

- Caché es un **mecanismo de almacenamiento temporal** que permite guardar datos de manera local para que se pueda acceder a ellos más rápidamente en futuras solicitudes.
- En lugar de solicitar la misma información repetidamente desde su fuente original (como un servidor), la caché almacena una copia de los datos que se usan con frecuencia, reduciendo el tiempo de acceso y la carga en el servidor.
- **Ejemplo de uso de la caché:**

Cuando se visita una página web, el navegador descarga el contenido, como imágenes y archivos CSS. Si se vuelve a visitar la misma página más tarde, en lugar de volver a descargar todo desde el servidor, **el navegador puede reutilizar los archivos almacenados en su caché local, lo que acelera el tiempo de carga.**

# Cache-Control

El encabezado "Cache-Control" se utiliza para controlar el almacenamiento en caché de las respuestas HTTP. Su función es especificar cómo los navegadores y los servidores proxy deben manejar el almacenamiento en caché de una respuesta.

- **max-age=N** La directiva indica que la respuesta se mantiene fresca hasta N segundos después de que se genere la respuesta.
- **no-cache** La directiva indica que la respuesta puede ser almacenada en cachés, pero la respuesta debe ser validada con el servidor de origen antes de cada reutilización, incluso cuando la caché está desconectada del servidor de origen.
- **must-revalidate** La directiva indica que la respuesta puede ser almacenada en cachés y puede ser reutilizada mientras esté fresca.
- **no-store**: La directiva indica que las cachés no deben almacenar esta respuesta.

# Cabeceras para gestionar el almacenamiento en caché

El encabezado **Expires** contiene la fecha y hora en la que se considerará la *respuesta caducada*.

El encabezado **Last-Modified** contiene la fecha y la hora en que el servidor de origen indica cuándo el recurso se modificó por última vez.

Se utiliza para determinar si un recurso recibido y el almacenado es el mismo.

Las páginas PHP a menudo generan contenido muy dinámico, y para evitar que los usuarios pierdan actualizaciones al ver las páginas almacenadas en la caché, es útil poder decirles a los navegadores que **no** almacenen en caché ciertas páginas o durante cuánto tiempo su contenido es válido.

```
<?php
header('Cache-Control:no-cache,no-store,must-revalidate');
header('Expires:Sun,01 Jul 2005 00:00:00 GMT');
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
?>
```





# HTTP Cookies

## ¿Qué es una cookie?

- Es información enviada o recibida en las cabeceras HTTP y que queda almacenada localmente en el lado del cliente(navegador) durante un tiempo determinado.
- Se envía hacia y desde el servidor web en las cabeceras HTTP.
- Como otros encabezados, cookies deben ser enviadas antes de cualquier salida en el script.

```
setcookie(  
    string $name,  
    string $value = "",  
    int $expires = 0,  
    string $path = "",  
    string $domain = "",  
    bool $secure = false,  
    bool $httponly = false  
): bool
```

- Esto requiere que hagas llamadas a esta función antes de cualquier salida que genere el script, incluyendo etiquetas <html> y <head> así como cualquier espacio en blanco.

## Otras funciones de php

- Retrasa la ejecución del programa durante el número de segundos

```
sleep(int $seconds): int
```

- Imprime un mensaje(opcional) y termina el script actual

```
exit(string $status = ?): void
```

```
die() – Equivalente a exit
```

```
exit("Fin del script");  
die("Fin del script");
```