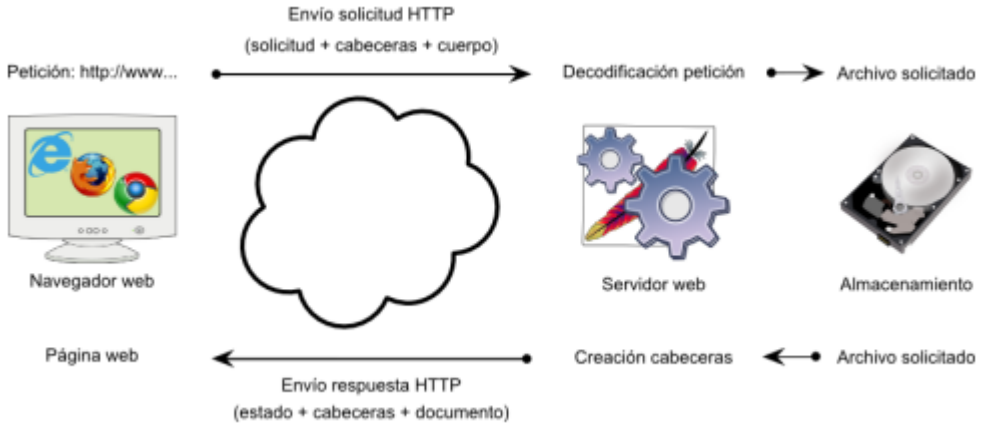
PHP

El protocolo HTTP

- ☐ Comunicación entre el cliente y el servidor
- ☐ Cuando un usuario solicita una página web a un servidor, el proceso se puede describir (de forma simplificada) en cuatro pasos:
  - ☐ El navegador solicita al servidor el documento mediante una solicitud HTTP.
  - ☐ El servidor prepara el documento.
  - ☐ El servidor envía el documento al navegador mediante una respuesta HTTP.
  - ☐ El navegador muestra el documento al usuario.



Solicitudes HTTP

- ☐ La solicitud HTTP está formada por varias líneas de texto:
  - La línea de solicitud, que incluye:
    - El método que se va a utilizar (GET, POST, TRACE, etc.).
    - La ubicación del documento solicitado (URI).
    - La versión del protocolo HTTP.

HTML:

```
<a href="https://example.com">Visitar ejemplo.com</a>
<form action="/submit" method="POST">
  <input type="text" name="nombre">
  <button type="submit">Enviar</button>
</form>
```

Solicitud HTTP (aproximada):

```
POST /submit HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
nombre=JohnDoe ( datos del formulario codificados en URL. )
```

  - Los campos de cabecera, que pueden incluir entre otros:
    - Referer: dirección de donde se ha obtenido la dirección del documento solicitado (si una página A contiene un enlace a otra página B, el navegador

solicita en la línea de petición la página B, pero también envía la dirección de la página A como referer).
        - User-agent: información sobre el navegador (nombre, versión, etc).
        - Accept: tipos MIME, juegos de caracteres, codificaciones, idiomas, etc. admitidos por el navegador.
        - Cookies: Si el servidor ha almacenado previamente cookies en el cliente, estas se incluyen en las peticiones.
    - Una línea en blanco.
    - El cuerpo del mensaje, que incluye texto opcional como por ejemplo:
        - datos de un formulario cuyo método sea POST

GET /pagina.html HTTP/1.1
Host: www.ejemplo.com
Referer: https://www.paginaanterior.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Cookie: session_id=123456789;
other_cookie=value
Connection: keep-alive

nombre=Juan&apellido=Pérez&edad=30

IMPORTANTE; **HTML no tiene un control directo sobre** las cabeceras HTTP como **`Referer, User-Agent o Cookie`. Estas** cabeceras **son** generalmente **agregadas automáticamente por el navegador.**

Respuestas HTTP

- ☐ **El servidor web** es el responsable de generar las respuestas HTTP.
    - ☐ Cuando un cliente (un navegador, por ejemplo) envía una solicitud, el servidor procesa la solicitud y envía una respuesta de acuerdo con las reglas del protocolo HTTP.
- ☐ Los factores que influyen en la respuesta HTTP son:
    - ☐ **La solicitud del cliente:** El método HTTP (GET, POST, PUT, DELETE), la URL, las cabeceras y el cuerpo de la solicitud.
    - ☐ **La configuración del servidor:** Las reglas de reescritura, los archivos de configuración, los scripts y aplicaciones que se ejecutan en el servidor.
    - ☐ **El contenido del servidor**
- ☐ La respuesta HTTP está formada por varias líneas de texto:
    - La línea de estado, que incluye:
        - La versión del protocolo HTTP.
        - El código de status (403, 404, 500, etc.).

    - El texto asociado al código de status (403=Forbidden, 404=Not found, 500=Internal Server Error, etc.).
- Los campos de cabecera, que pueden incluir entre otros:
    - Location:permite decirle al cliente que solicite otro documento en lugar del documento solicitado inicialmente.
    - Content: tipo MIME, juego de caracteres, codificación, idioma, etc. del documento enviado.
    - Set-Cookie: permite decirle al cliente que cree una cookie.
- Una línea en blanco.
- El cuerpo del mensaje, que incluye el documento solicitado por el cliente.

```
HTTP/1.1 302 Found
Location: https://www.nuevositio.com/
Content-Type: text/html; charset=utf-8

<html>
<head>
  <meta http-equiv="Refresh" content="0; URL=https://www.nuevositio.com/">
</head>
<body>
  Redireccionando...
</body>
</html>
```

Cabeceras: la función header()

Función header()

- ☐ Cuando un **servidor envía** una página web al navegador envía **la página web e información adicional** (*el estado y los campos de cabecera*).
    - ☐ Tanto el **estado** como los **campos de cabecera** se envían a**ntes de la página web.**
    - ☐ *Normalmente*, **un programa PHP sólo genera la página web** y **es el servidor el que genera** automáticamente la **información de estado y los campos de cabecera** y los envía antes de enviar el contenido generado por el programa.
    - ☐ Pero **un programa PHP** también **puede generar la información de estado y los campos de cabecera**, mediante la **función header().**

Ejemplo

```php
<?php
// Configurar la cabecera de estado HTTP
header('HTTP/1.1 200 OK');

// onfigurar las cabecCeras personalizadas
header('Content-Type: text/html; charset=utf-8');
header('X-Powered-By: MiFramework PHP');
```

```php
// Enviar una cookie
setcookie('nombre_cookie', 'valor_cookie', time() + 3600); // Expira en
1 hora

// Generar el contenido HTML
echo '<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de cabeceras en PHP</title>
</head>
<body>
  <h1>¡Hola desde mi servidor PHP!</h1>
  <p>Esta página ha sido personalizada con cabeceras HTTP.</p>
</body>
</html>';
?>
```

Explicación del código:

Configuración del estado HTTP:
header('HTTP/1.1 200 OK');: Establece el código de estado HTTP en 200 (OK), indicando que la solicitud se ha completado con éxito.

Configuración de cabeceras personalizadas:
header('Content-Type: text/html; charset=utf-8');: Especifica que el contenido de la respuesta es HTML y utiliza la codificación UTF-8.
header('X-Powered-By: MiFramework PHP');: Agrega una cabecera personalizada para indicar que la página fue generada por tu propio framework PHP.

Envío de una cookie:
setcookie('nombre_cookie', 'valor_cookie', time() + 3600);: Crea una cookie llamada *nombre_cookie* con el valor *valor_cookie* que expirará en una hora.

Generación del contenido HTML:
Se crea el contenido HTML de la página y se envía al navegador.

En access.log de APACHE podemos ver el estado de la solicitud HTTP y cabecera. En este caso :
"GET /clase/Propio/Cabecera.php HTTP/1.1" 200 219 "http://localhost/clase/Propio/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36"

Puntos clave a recordar:

- ☐ **Orden**: Las llamadas a **header() deben realizarse antes de cualquier otro contenido HTML o PHP**, ya que una vez que se envía cualquier salida, no se pueden enviar más cabeceras.
- ☐ **Codificación**: Utilizar la codificación correcta para evitar problemas de visualización de caracteres especiales.
- ☐ **Cookies**: Las cookies se utilizan para almacenar información en el lado del cliente.
- ☐ **Cabeceras personalizadas**: Puedes definir tus propias cabeceras para transmitir información específica a tu aplicación o a terceros.

El ejemplo siguiente muestra un ejemplo de cabecera HTTP, concretamente una redirección. En el ejemplo, la página 1 contiene un enlace a la página 2. Pero como la página genera una cabecera HTTP de redirección a la página 3, al hacer clic en el enlace se muestra directamente la página 3.

cabeceras-header-1-1.php

```
<p>Esta es la página 1.</p>
<p><a href="cabeceras-header-1-2.php">Enlace a la página 2 (que
redirige a la página 3)</a></p>
```

cabeceras-header-1-2.php

```
<?php
header("Location:cabeceras-header-1-3.php");
print "<p>Esta es la página 2</p>";
print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";
print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
?>
```

cabeceras-header-1-3.php

```
<p>Esta es la página 3.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página

1</a></p>
```

Nota: **Si el programa contiene un error antes de la función** header(), PHP generará un aviso de error que es también parte de la salida del programa y eso **provocaría el fallo de la función header() posterior.**

Redirecciones: header("Location:...")

- ☐ La función **header()** se puede utilizar para **redirigir automáticamente a otra página**, enviando como **argumento** la **cadena Location**: *seguida de la dirección absoluta o relativa de la página* a la que queremos redirigir.
- ☐

Ejemplos,
Redireccionamiento tras un formulario: ***Un usuario llena un formulario de contacto y al enviar, se le redirige a una página de agradecimiento.***

Si el formulario se envía (método POST), el usuario es redirigido a la página gracias.php. Esto evita que el usuario vuelva a enviar el formulario accidentalmente al recargar la página.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Procesar los datos del formulario
    // ...

    header("Location: gracias.php");
    exit;
}
```

Redireccionamiento después de un proceso: ***Un usuario inicia sesión y es redirigido al panel de control.***

Si las credenciales son correctas, el usuario es redirigido al panel de control.

```
if (// Verificar credenciales) {
    // Iniciar sesión
    $_SESSION['usuario'] = $usuario;
    header("Location: panel_control.php");
    exit;
} else {
    // Mostrar mensaje de error
}
```

3. Redireccionamiento 301 para SEO:Se ha cambiado la URL de una página y se quiere redirigir permanentemente el tráfico antiguo a la nueva URL.

Explicación: El código 301 indica a los motores de búsqueda que la página se ha movido de forma permanente, ayudando a mantener el SEO.

```
header("HTTP/1.1 301 Moved Permanently");
header("Location: https://nuevo-dominio.com/nueva-pagina");
exit;
```

Redireccionamiento condicional:Redirigir a usuarios no registrados a una página de registro.

Si el usuario no está autenticado, se le redirige a la página de registro.

```
if (!isset($_SESSION['usuario'])) {
    header("Location: registro.php");
    exit;
}
```

Redireccionamiento basado en el navegador; Este código redirige a los usuarios móviles a una versión optimizada para móviles de la página.

Obtiene el agente de usuario del navegador del visitante. Y lo redirige a la página optimizada para su navegación.

```
$user_agent = $_SERVER['HTTP_USER_AGENT'];
if (strpos($user_agent, 'Mobile') !== false) {
  header('Location: version_movil.php');
  exit;
} else {
  header('Location: version_escritorio.php');
  exit;
}
```

- ☐ **Si** además de redirigir a una página, **se quieren enviar controles a dicha página,** se pueden **añadir a la cadena separando los controles con el carácter &.**

cabeceras-header-1-2.php

```
<¿<?php

header('Location:cabeceras-header-1-3.php?busqueda=coches&categoria=dep
ortivos&pagina=2');
print "<p>Esta es la página 2</p>";
print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";
print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
?>
```

cabeceras-header-1-3.php

```php
<?<?php
print_r($_REQUEST);
print_r(nl2br("\n"));
print_r($_GET);
echo "<p>Esta es la página 3.</p>
<p>La redirección <strong>SÍ</strong> se ha realizado.</p>";
header('cabeceras-header-1-1.php');
echo "<p>Volver a la página 1</p>";

?>
```

- ☐ **Si el valor a enviar contiene espacios, se pueden escribir caracteres + o espacios para separar las palabras:**

```php
<?php
header("Location:cabeceras-header-1-3.php?nombre=Pepe+Sol&edad=25
");
?>
```

Resto del programa tras la redirección

- ☐ **La ejecución de un programa no se detiene al encontrar una redirección**, sino que PHP ejecuta el programa hasta el final. **Dependiendo de las instrucciones que haya tras la dirección, el resultado puede ser relevante o no.**
- ☐ **Si se escriben varias redirecciones, se aplicará la última.** En el ejemplo siguiente, la página redirigirá a “cabeceras-header-1-4.php”.

cabeceras-header-1-2.php

```php
<?php
header("Location:cabeceras-header-1-3.php");
header("Location:cabeceras-header-1-4.php");
print "<p>Esta es la página 2</p>";
print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";
print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
?>
```

cabeceras-header-1-4.php

```php
<?php
print_r($_REQUEST);
print_r(nl2br("\n"));
print_r($_GET);
```

```
echo "<p>Esta es la página 4.</p>
<p>La redirección <strong>SÍ</strong> se ha realizado.</p>";
//header('Location:cabeceras-header-1-1.php');
?>
<a href="cabeceras-header-1-1.php"> ir a otra página </a>
```

- **Si después de la redirección se genera texto,** ese texto **no llegará al usuario**, puesto que la redirección le llevará antes a otra página
- **Si el programa realiza acciones como modificar registros en una base de datos, borrar archivos, etc.,** esas acciones **sí que se realizan**.
- Si tras una redirección no queremos que se ejecute el resto del programa, tenemos dos opciones:
  - Podemos utilizar una expresión if ... else …
  - Podemos utilizar la instrucción exit, que detiene el programa en ese punto
    - **exit no es una función,** sino una palabra reservada del lenguaje, **pero también se puede utilizar** seguida de paréntesis **como una función:** exit().
      - Se puede incluso incluir un parámetro que puede ser un entero entre 0 y 255 o una cadena:
        - Si el parámetro es un valor numérico, este valor se utilizará como estado de salida y no se imprimirá. Los estados de salida deben estar en el rango de 0 a 254, el estado de salida 255 está reservado por PHP y no se utilizará. El estado 0 se utiliza para finalizar el programa con éxito.
        - Si el valor es una cadena, se imprime su valor antes de terminar.

- Pero **hay que tener en cuenta que** exit **detiene no sólo la ejecución del programa, sino también la de los programas que hubieran llamado a esos programas**

EJEMPLO CON IF

```
<?php
        if (condicion) {
        header("Location:https://www.google.com");
} else {
        …
}
?>
```

EJEMPLO CON EXIT
(cabeceras-header-1-2.php)

```
<?php
        header("Location:cabeceras-header-1-3.php");
        exit;
        header("Location:cabeceras-header-1-4.php");
```

```
        print "<p>Esta es la página 2</p>";
        print "<p>La redirección <strong>NO</strong> se ha
        realizado</p>";
        print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
        1</a></p>";
?>
```

Crear con PHP otros tipos de archivos

- ☐ Con instrucciones print, 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.**
- ☐ Existen muchos tipos MIME (véase lista de tipos MIME en MDN)

extensión del archivo	tipo de archivo	tipo MIME
.html	página web HTML	text/html
.css	hoja de estilo CSS	text/css
.js	JavaScript	application/js
.svg	SVG	image/svg+xml
.json	JSON	application/json

- ☐ **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.**

Ejemplo

Cuando se ejecuta este script, el navegador recibirá un archivo de texto llamado "ejemplo.txt" con el contenido especificado. El navegador interpretará el contenido como texto plano y lo mostrará o permitirá al usuario guardarlo.

```php
<?php
// Generamos un archivo de texto simple
$texto = "Este es el contenido de un archivo de texto generado por
PHP.";

// Definimos el tipo MIME como texto plano
```

```
header('Content-Type: text/plain');

// Indicamos al navegador que descargue el archivo con un nombre específico
header('Content-Disposition: attachment; filename="ejemplo.txt"');

// Imprimimos el contenido del archivo
print $texto;
?>
```

Crear hojas de estilo CSS

- ☐ Para que el navegador aplique la hoja de estilo, e**l programa debe generar una cabecera que declare el tipo MIME de hoja de estilos text/css**, como muestra el siguiente ejemplo. CUIDADO ;*el servidor asigna por defecto el tipo MIME text/html a los ficheros generados por un programa PHP*

```
<link rel="stylesheet" href="mime-css.php" title="Color">
```

mime-css.php

```
<?php
header("Content-type: text/css");
$c1 = rand(0, 255);
$c2 = rand(0, 255);
$c3 = rand(0, 255);
print "body{";
print " background-color: rgb($c1, $c2, $c3);";
print " color: rgb($c2, $c3, $c1);";
print "}";
?>
```

Crear imágenes SVG

- ☐ El programa siguiente crea las etiquetas correspondientes a una imagen SVG, concretamente a un cuadrado de color aleatorio. Si escribimos directamente la url de ese programa en el navegador, el navegador muestra el contenido esperado.

```
<?php
$c1 = rand(0, 255);
$c2 = rand(0, 255);
$c3 = rand(0, 255);
print "<svg version=\"1.1\"
xmlns=\"http://www.w3.org/2000/svg\" " . " width=\"100\"
height=\"100\" viewBox=\"-5 -5 100 100\">";
```

```
print " <rect fill=\"rgb($c1, $c2, $c3)\" x=\"0\" y=\"0\"
width=\"90\" height=\"90\" />";
print "</svg>";
?>
```

- **Aunque el navegador consiga mostrar el cuadrado, realmente no estamos haciendo bien las cosas,** en otras circunstancias, este mismo programa no daría el resultado esperado.
  - Como muestra el siguiente ejemplo, **si el programa se llama en una etiqueta** , el navegador no puede mostrar la imagen. El problema es que en una etiqueta **el navegador tiene que recibir obligatoriamente algún tipo MIME de imagen** (jpg, png, svg, etc.).
  -

```
<p><img src="svg-sin-mime.php" alt="ERROR No puedo mostrar la
imagen"></p>

<?php
        $c1 = rand(0, 255);
        $c2 = rand(0, 255);
        $c3 = rand(0, 255);
        print "<svg version=\"1.1\"
        xmlns=\"http://www.w3.org/2000/svg\" " . " width=\"100\"
        height=\"100\" viewBox=\"-5 -5 100 100\">";
        print " <rect fill=\"rgb($c1, $c2, $c3)\" x=\"0\" y=\"0\"
        width=\"90\" height=\"90\" />";
        print "</svg>";
?>
```

- **Una imagen SVG debería entregarse al navegador con el tipo MIME image/svg+xml.**

```
<p><img src="svg-con-mime.php" alt="ERROR No puedo mostrar la
    imagen"></p>
```

```
<?php
header("Content-type: image/svg+xml");
$c1 = rand(0, 255);
$c2 = rand(0, 255);
$c3 = rand(0, 255);
print "<svg version=\"1.1\"
xmlns=\"http://www.w3.org/2000/svg\" " . " width=\"100\"
height=\"100\" viewBox=\"-5 -5 100 100\">";
```

```
print " <rect fill=\"rgb($c1, $c2, $c3)\" x=\"0\" y=\"0\"
width=\"90\" height=\"90\" />";
print "</svg>";
?>
```