

DESARROLLO WEB EN ENTORNO SERVIDOR

UNIDAD 4. DESARROLLO DE APLICACIONES CON CÓDIGO EMBEBIDO

1. CABECERAS HTTP

Como se explica en la unidad 1, el protocolo HTTP envía cabeceras de información junto con las peticiones que mandan los clientes y las respuestas del servidor. La información contenida en estas cabeceras se puede consultar con las siguientes funciones de PHP:

- `apache_request_headers()`: devuelve un vector asociativo que contiene los campos y los valores de la cabecera de petición que ha enviado el cliente.
- `apache_response_headers()`: devuelve un vector asociativo que contiene los campos y los valores de la cabecera de respuesta que envía el servidor al cliente.

También existen dos funciones para la gestión de la cabecera de respuesta que envía el servidor al cliente:

- `header(cadena, reemplazo, código)`: envía una cabecera de respuesta al cliente con la información indicada en *cadena*. Opcionalmente, se puede indicar si se reemplaza la cadena a enviar por la especificada (`true`, valor por defecto) o se añade una cabecera del mismo tipo (`false`). También se puede indicar, opcionalmente, un código HTTP de respuesta de tres cifras (véase una referencia completa de estos códigos en https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP).
- `headers_sent()`: devuelve un valor `true` o `false` indicando si la cabecera de respuesta ha sido enviada correctamente al servidor.

Las cabeceras de los mensajes HTTP enviados desde el servidor permiten cambiar el comportamiento del navegador del cliente en algunas situaciones. Por ejemplo, se puede especificar el tipo de información enviada, la redirección a otras páginas o el control de la caché. La **caché del navegador** se utiliza para almacenar páginas visitadas con anterioridad, de forma que, al volver a consultarlas no es necesario tener que descargar toda la información de nuevo, lo que agiliza la navegación y reduce la cantidad de datos transferidos. Sin embargo, tiene el inconveniente de no mostrar los cambios realizados en páginas recientemente modificadas.

Es importante tener en cuenta que se producirá un error si se ejecuta la función `header()` cuando la cabecera de respuesta ya ha sido enviada al cliente. Por lo tanto, esta función debe ser llamada antes de que esto haya ocurrido, que sucede cuando, mediante `echo`, `print` o `printf()`, se genera salida que llena el espacio de almacenamiento temporal del mensaje de respuesta enviado al cliente (directiva `output_buffering`, con el valor por defecto de 4096 bytes). La función `header()` se puede utilizar para lo siguiente:

- `header('Location: ruta_nueva')`: sirve para redirigir el navegador cliente a otra página. En *ruta_nueva* se especifica cualquier página HTML o PHP, además de cualquier dirección de Internet.
- `header('refresh:tiempo;url=ruta_nueva')`: también sirve para redirigir el navegador cliente a otra página, después de que haya pasado el tiempo indicado. En *ruta_nueva* se especifica cualquier página HTML o PHP, además de cualquier dirección de Internet. A continuación de esta función, se puede mostrar un mensaje en la página indicando al usuario que se va a redirigir a otro lugar.
- `header('Content-Type: tipo/subtipo')`: especifica el tipo de contenido que el servidor va a enviar al cliente. Se pueden indicar los siguientes tipos principales con los siguientes subtipos: `text` (`plain`, `html` o `xml`), `multipart` (`form-data` o `digest`), `message`

(partial o rfc822), image (png, gif o jpg), audio (mp3, wma o 32kadpcm), video (mpeg, avi o mp4) o application (json o pdf).

- header ('Expires: fecha_y_hora'): especifica la fecha y hora de expiración de la caché del navegador. La fecha y hora se especifica en formato "%a, %d %b %Y %H:%M:%S GMT".
- header ('Last-Modified: fecha_y_hora'): especifica la fecha y hora de la última modificación de la página solicitada, en formato "D, d M Y H:i:s GMT".
- header ('Cache-Control: parámetros'): establece los parámetros de control de la caché del navegador, separados por comas. En HTTL 1.1 pueden ser: public (caché pública), private (caché privada), no-cache (no hay caché) y no-store (hay caché, pero no se archiva). Una lista completa se puede encontrar en: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control>.
- header ('Pragma: no-cache'): se usa para especificar que no hay caché. Si se indica, debe especificarse también 'Cache-control: no-cache'.

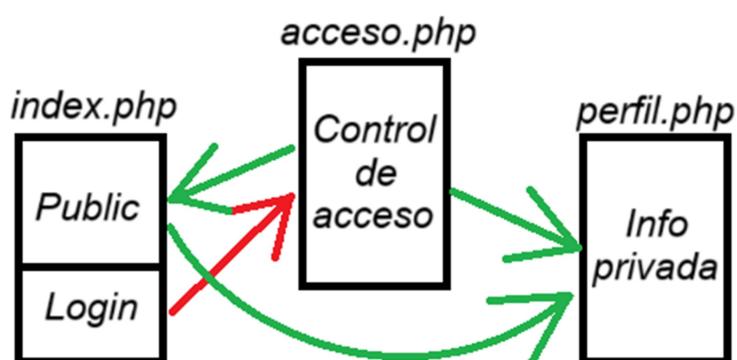
2. MANTENIMIENTO DEL ESTADO DE UNA CONEXIÓN

El protocolo HTTP es de tipo **stateless** (sin estado), esto significa que, cada vez que un mismo cliente solicita una página diferente a un servidor web, estas solicitudes son independientes y no hay ninguna información común entre ellas. De esta forma, el intercambio de información entre el cliente y el servidor se compone de varias comunicaciones independientes, cada una con su solicitud y su respuesta. Los protocolos de este tipo hacen que los servidores no almacenen ninguna información del estado de las comunicaciones con cada cliente. Por el contrario, los protocolos que permiten mantener información sobre el estado de las conexiones se llaman **statefull** (con estado).

El problema principal de HTTP radica en que, cuando queremos diseñar una web que almacene información privada de los usuarios, necesitamos establecer algún mecanismo que haga dos cosas:

- Permite la identificación de cada usuario, con un nombre y una contraseña.
- La identificación sea recordada durante todo el tiempo en que el usuario navegue por todas las páginas del sitio, sean privadas o no y no sea necesario tener que volver a introducir estas credenciales en cada página del sitio que es visitada.

Supongamos que tenemos la siguiente estructura de nuestro sitio web, en la que tenemos una página principal llamada index.php (donde aparece información pública y un formulario de acceso) y una página llamada acceso.php (donde se comprueba si el usuario y la contraseña introducidas son correctas). El proceso de autenticación está indicado con la flecha roja y, una vez realizado, el usuario puede navegar siguiendo las flechas verdes, de nuevo a la página pública principal (donde ya no le aparecerá el formulario de acceso) o podrá visitar la página perfil.php, con información confidencial.



Puesto que HTTP es *stateless*, una vez que el usuario se ha identificado correctamente (en la página acceso.php), no hay ningún mecanismo que permita almacenar esta información para recordar que la

identificación se ha realizado correctamente. En esta situación ¿cómo se puede navegar por las páginas index.php y perfil.php sin que el usuario tenga que volver a identificarse?

Este problema no sólo se produce cuando los usuarios se identifican en algún sitio web, sino que también hay que tratarlo cuando un usuario establece unas determinadas preferencias personalizadas, disponibles en algunas páginas. Por ejemplo, recordar qué búsquedas realizó con anterioridad, qué productos visitó en una tienda, el criterio de ordenación en una lista de ítems, etc., son preferencias que deben almacenarse en algún lugar para que estén establecidas cuando se vuelva a visitar la misma página.

Sin embargo, existen dos mecanismos que permiten almacenar información entre diferentes peticiones de un mismo cliente: las **cookies**, que se almacenan en el navegador del cliente y las **sesiones**, que se almacenan en el lado del servidor web.

2.1. COOKIES

Las cookies se almacenan desde PHP en el vector global `$_COOKIE`. Lo que se guarde dentro de este vector, se guardará en el navegador del cliente de forma automática. Sin embargo, hay que tener en cuenta que el cliente puede configurar su navegador para no almacenar estas cookies. Los navegadores también tienen limitación en cuanto al tamaño de cada cookie (máximo 4 Kb), a la cantidad de cookies que se pueden guardar para cada sitio web visitado y la cantidad total para todos los sitios.

En PHP, para crear una cookie se utiliza la función `setcookie()`, que tiene dos sintaxis diferentes, solamente el parámetro nombre es obligatorio:

- `setcookie(nombre, valor, expira, ruta, dominio, seguro, httponly)`: en `expira` se indica la fecha de eliminación (el valor 1 se usa para eliminarla inmediatamente, para eliminarla dentro de un tiempo se puede usar `time() + tiempo` en segundos).
- `setcookie(nombre, valor, opciones)`: igual que la anterior (`opciones` es un vector).

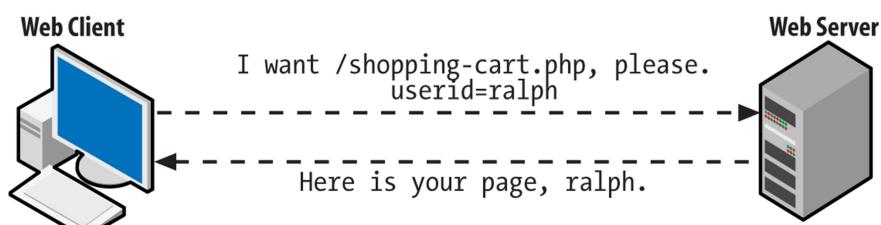
El cliente puede ver las cookies que están almacenadas en su navegador, por ejemplo, en Firefox se hace seleccionando en el menú principal **Ajustes->Privacidad y Seguridad->Cookies y datos del sitio->Administrar datos**. En Chrome se pueden consultar en el menú principal **Configuración->Privacidad y Seguridad->Configuración de sitios->Cookies de terceros->Ver todos los datos y permisos del sitio**.

La figura siguiente muestra el esquema de comunicación entre un cliente y un servidor, además del momento en el que éstos se intercambian la información de las cookies.

First Request



Second Request



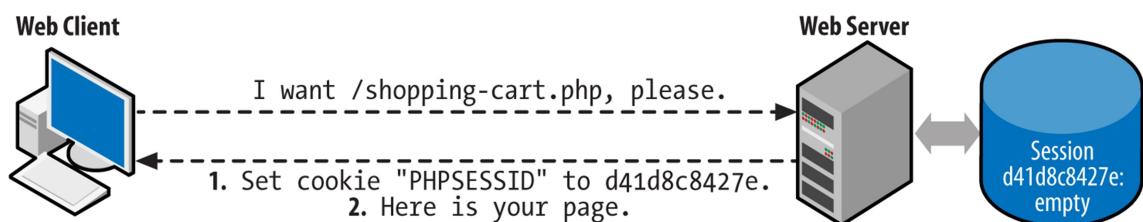
Las cookies se pueden utilizar para muchos propósitos, en general, para guardar información sobre los clientes. Entre estos propósitos, podemos encontrar: recordar los inicios de sesión, almacenar valores temporales de usuario, guardar las preferencias de ordenación de una lista de elementos,

En el lado del cliente, existen alternativas a las cookies para almacenar información, como son los objetos `LocalStorage` y `SessionStorage` de JavaScript.

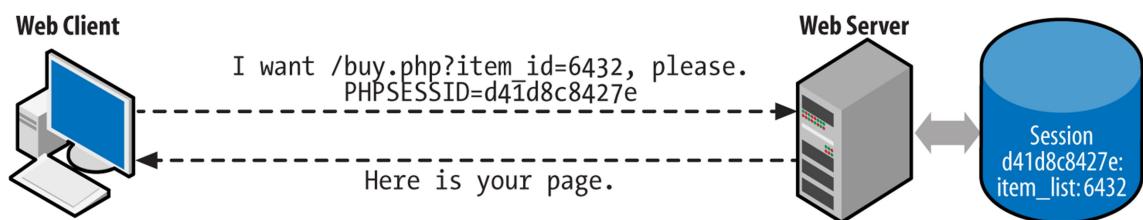
2.2. SESIONES

Las sesiones permiten añadir la gestión del estado al protocolo HTTP, almacenando en este caso la información en el servidor. Cada visitante tiene un identificador de sesión único (*id*) el cual, por defecto, se almacena en una cookie, llamada por defecto `PHPSESSID`, en el lado del cliente. Si el cliente no tiene las cookies activadas, el identificador se propaga en cada dirección (URL) dentro del mismo dominio. Cada sesión tiene asociado un almacén de datos, mediante el vector global `$_SESSION`, en el cual se puede almacenar y recuperar información.

First Request



Second Request



Las operaciones que podemos realizar con una sesión son las siguientes:

- `session_name()`: devuelve el nombre de la sesión, su valor por defecto es `PHPSESSID`.
- `session_name(nombre)`: establece un nombre para la sesión, que también será el nombre de la cookie asociada. Debe hacerse antes que `session_start()` o producirá un error.
- `session_start()`: inicia la sesión.
- `session_id()`: devuelve el identificador de sesión (*id*). Es el mismo valor que se guarda en la cookie del cliente.
- `$_SESSION[clave]=valor`: establece un valor para una clave de la sesión.
- `$_SESSION[clave]`: devuelve el valor de la clave de la sesión.
- `unset($_SESSION[clave])`: elimina la clave de la sesión y su valor.
- `session_destroy()`: destruye la sesión.

Las siguientes directivas permiten configurar algunos aspectos de la sesión, éstas se pueden consultar con `phpinfo()` o `ini_get()`, y también se pueden modificar en `php.ini` o con `ini_set()`:

- `session.save_handler`: indica cómo se van a guardar los datos de la sesión. El valor por defecto es `files` (en archivos), para más detalles véase `session_set_save_handler()`.
- `session.save_path`: ruta donde se almacenan los archivos con los datos de la sesión. El valor por defecto es `"/tmp"`.

- `session.name`: especifica el nombre de la sesión si no se indica ninguno con `session_name()`. Su valor por defecto es `PHPSESSID`.
- `session.auto_start`: especifica si la sesión se inicia automáticamente en cada página. El valor por defecto es 0 (deshabilitado).
- `session.cookie_lifetime`: establece el tiempo de vida en segundos de la cookie de la sesión, si no se especifica con la función `session_set_cookie_params(tiempo)`. El valor por defecto es 0 (hasta que se cierre el navegador). Se puede indicar lo siguiente para establecer un tiempo de inactividad: `setcookie(session_name(), session_id(), time() + $t)`.

3. AUTENTIFICACIÓN DE USUARIOS

El control de la autenticación de los usuarios se puede realizar mediante cookies o mediante la sesión. Sin embargo, el método más utilizado, por ser más seguro y por no depender de la configuración del cliente, es a través de la sesión. Los pasos que se realizan son los siguientes:

1. Se muestra la página con el formulario de acceso para escribir el usuario y la contraseña.
2. Los datos de autenticación son enviados al servidor, que los recoge en una página donde se crea la cookie o la sesión y se comprueba si el usuario y la contraseña son válidos.
3. Si son válidos, se añade una clave a la cookie o a la sesión para saber que se ha autenticado.
4. Si el usuario accede a cualquier página privada, lo primero que hay que hacer en ella es comprobar que existe la clave de la cookie o de la sesión, lo que asegura que el usuario se ha autenticado. Si no existe, no debe poder acceder a esta página (lo normal es que se redirija a una página pública o al formulario de acceso).
5. Cuando el usuario desea salir de la parte privada del sitio y pulsa en el botón correspondiente, hay que destruir la cookie o la sesión.

Cuando se realiza una web con varias páginas que se enlazan unas con otras, es recomendable dibujar un diagrama de navegación, para conocer qué enlaces debe poner en cada una de ellas y a dónde debe redirigirlas o qué páginas van a recoger la información enviada por los clientes en los formularios.

4. MANEJO DE ERRORES

Como se explicó en la unidad 1, PHP puede mostrar diferentes tipos de errores en la página, los más importantes son: las notificaciones (identificadas por la constante `E_NOTICE`), los avisos (identificados por la constante `E_WARNING`) y los errores graves (identificados por la constante `E_ERROR`). En el siguiente enlace se puede encontrar una descripción completa de todos los tipos de errores:

<https://www.php.net/manual/es/errorfunc.constants.php>

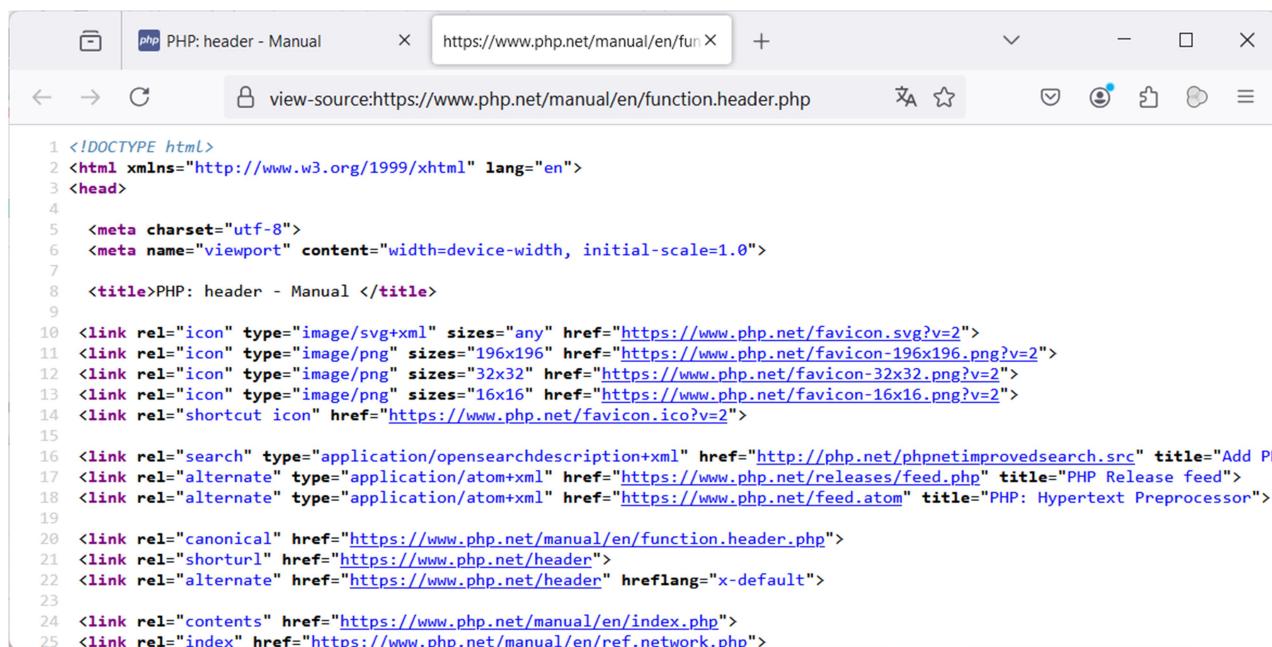
A la hora de indicar si se muestran estos tipos de errores en las páginas, se usan las directivas `display_errors` y `error_reporting` de `php.ini`. También se pueden usar las funciones `ini_set('display_errors', n)` para mostrar (`n` a 1) o no mostrar (`n` a 0) los errores y `error_reporting(códigos_errores)` para indicar qué tipos de errores mostrar (se indican los tipos de errores separados por `|`).

Se puede usar la función `set_error_handler(función)` para especificar otra función que se ejecutará cada vez que se produzca un error (se hará aunque se haya usado previamente `error_reporting()` para ocultar los errores). Esta segunda función debe ser definida con los siguientes parámetros: `$errno` (número de nivel del error), `$errstr` (cadena con el mensaje de error), `$errfile` (cadena con el nombre del archivo donde se ha producido el error), `$errline` (número de línea donde se ha producido) y `$errcontext` (vector con una tabla de símbolos del punto del error). Los siguientes errores no pueden ser manejados por esta función definida: `E_ERROR`, `E_PARSE`, `E_CORE_ERROR`, `E_CORE_WARNING`, `E_COMPILE_ERROR`, `E_COMPILE_WARNING`.

También se puede utilizar el operador de control de errores @ antes de una expresión que devuelva un valor, que se usa para omitir cualquier error que se pueda producir al evaluar dicha expresión. El error no será mostrado, pero si se ha definido una función con `set_error_handler()`, sí que se llamará. Hay que tener cuidado con este operador, ya que se pueden ocultar errores que luego son difíciles de depurar. A partir de PHP 8.0, este operador no silencia los siguientes errores: `E_ERROR`, `E_CORE_ERROR`, `E_COMPILE_ERROR`, `E_USER_ERROR`, `E_RECOVERABLE_ERROR` ni `E_PARSE`. Entre los programadores, este operador es conocido como STFU o “shut-up” (silencio).

5. HERRAMIENTAS DE DEPURACIÓN DE LOS NAVEGADORES

Los navegadores modernos disponen de un conjunto de herramientas a disposición de los programadores que permiten consultar información más completa sobre la web que se está desarrollando. La herramienta básica es la inspección del código fuente, que muestra el código HTML de la página tal y como ha sido generado por el servidor y enviado al cliente.



The screenshot shows a browser window with the URL `https://www.php.net/manual/en/function.header.php`. The title bar says "PHP: header - Manual". The page content is the source code of the PHP documentation page, which includes HTML tags like `<html>`, `<head>`, and `<body>`, along with various `<link>` tags for favicons and other resources.

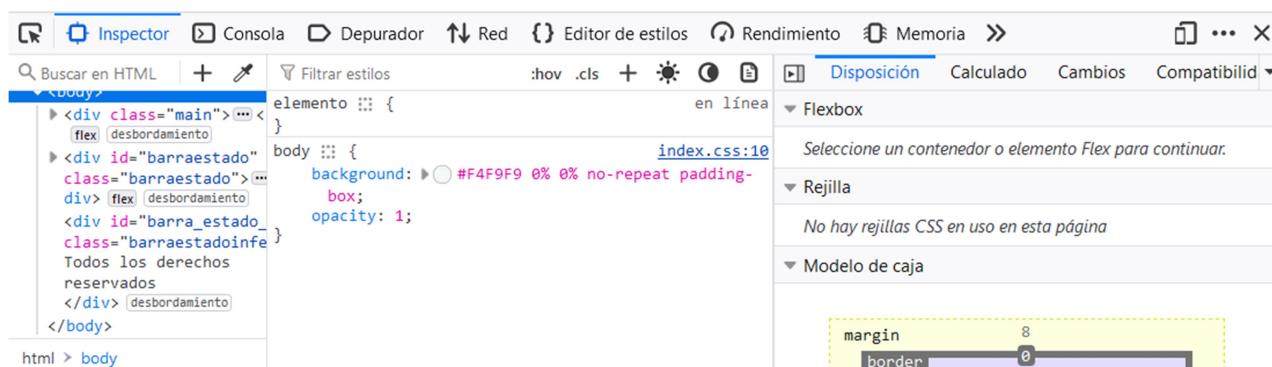
```

1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
3 <head>
4
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7
8   <title>PHP: header - Manual </title>
9
10  <link rel="icon" type="image/svg+xml" sizes="any" href="https://www.php.net/favicon.svg?v=2">
11  <link rel="icon" type="image/png" sizes="196x196" href="https://www.php.net/favicon-196x196.png?v=2">
12  <link rel="icon" type="image/png" sizes="32x32" href="https://www.php.net/favicon-32x32.png?v=2">
13  <link rel="icon" type="image/png" sizes="16x16" href="https://www.php.net/favicon-16x16.png?v=2">
14  <link rel="shortcut icon" href="https://www.php.net/favicon.ico?v=2">
15
16  <link rel="search" type="application/opensearchdescription+xml" href="http://php.net/phpnetimprovedsearch.src" title="Add PH
17  <link rel="alternate" type="application/atom+xml" href="https://www.php.net/releases/feed.php" title="PHP Release feed">
18  <link rel="alternate" type="application/atom+xml" href="https://www.php.net/feed.atom" title="PHP: Hypertext Preprocessor">
19
20  <link rel="canonical" href="https://www.php.net/manual/en/function.header.php">
21  <link rel="shorturl" href="https://www.php.net/header">
22  <link rel="alternate" href="https://www.php.net/header" hreflang="x-default">
23
24  <link rel="contents" href="https://www.php.net/manual/en/index.php">
25  <link rel="index" href="https://www.php.net/manual/en/ref.network.php">

```

5.1. HERRAMIENTAS DE MOZILLA FIREFOX

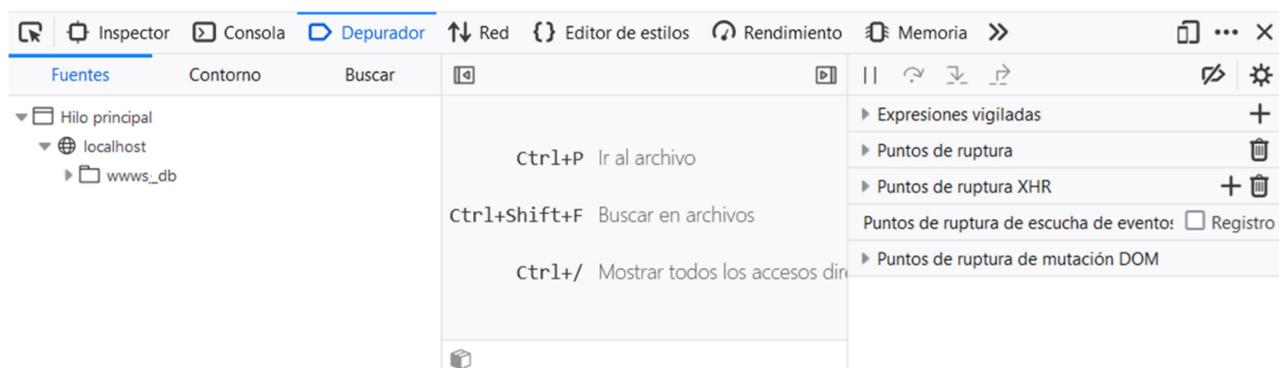
Desde el navegador de Mozilla Firefox, se puede acceder a las herramientas de depuración seleccionando en el menú principal **Más herramientas->Herramientas para desarrolladores**. La pestaña de más a la izquierda se llama **Inspector** y contiene todos los elementos de la página HTML (a la izquierda) con sus atributos CSS (en el centro). Se pueden desactivar algunos atributos y el aspecto de la página cambiará.



La siguiente pestaña es la **Consola**, en ella aparecen todos los errores que se producen cuando el navegador interpreta la página (tanto errores HTML como JavaScript).



La siguiente pestaña es el **Depurador**, que permite parar la carga de la página en el navegador, estableciendo puntos de ruptura. También se pueden incluir expresiones que serán evaluadas para comprobar si las variables tienen los valores adecuados.



En la pestaña **Red**, se muestran todos los archivos que se cargan con la página (hojas de estilo, imágenes, etc.). Si se selecciona una página HTML, se pueden inspeccionar las cabeceras y las cookies que se envían junto con ella (figura de más abajo).

A screenshot of the Firefox Developer Tools interface, specifically the 'Red' (Network) tab. The tab bar includes 'Inspector', 'Console', 'Depurador', 'Red' (selected), 'Editor de estilos', 'Rendimiento', 'Memoria', and 'More'. The main pane shows a table of network requests. The columns include Estado (Status), Método (Method), Dominio (Domain), Archivo (File), Iniciador (Initiator), Tipo (Type), Transferido (Transfered), Tamaño (Size), and Duración (Duration). Requests listed include: GET /index.php?m=1 HTTP/2 (status 200), GET /index.php?m=2 HTTP/2 (status 200), GET /index.php?m=3 HTTP/2 (status 200), GET /index.php?m=4 HTTP/2 (status 200), and several image requests like favicon.ico and various .webp files. At the bottom, it shows 13 solicitudes (13 requests), 1.10 MB / 34.82 KB transferido (1.10 MB / 34.82 KB transferred), Finalizado: 10,11 s (Completed: 10,11 s), DOMContentLoaded: 53 ms (DOMContentLoaded: 53 ms), and load: 106 ms (load: 106 ms).

A screenshot of the Firefox Developer Tools interface, specifically the 'Red' (Network) tab. The tab bar includes 'Inspector', 'Console', 'Depurador', 'Red' (selected), 'Editor de estilos', 'Rendimiento', 'Memoria', and 'More'. The main pane shows a table of network requests. A specific request for '/pagead/interaction/?' is selected, showing its details. The 'Cabeceras' (Headers) tab is active, displaying the following response headers: access-control-allow-origin: *, alt-svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000, cache-control: no-cache, must-revalidate, and content-length: 0. There are also buttons for Bloquear (Block) and Reenviar (Rewrite).

The screenshot shows the Firefox Developer Tools Network tab. It lists several network requests, including JavaScript and CSS files, and their corresponding responses. A specific cookie named 'test_cookie' is highlighted in the 'Cookies' section of the response details. The cookie has the following properties:

- domain: ".doubleclick.net"
- expires: "2024-09-26T18:53:54.000Z"
- httpOnly: true
- path: "/"

En la pestaña titulada **Editor de estilos** se puede examinar el código CSS que se aplica a los diferentes elementos de la página.

The screenshot shows the Firefox Developer Tools Style tab. On the left, a sidebar lists several CSS files with their rule counts: 'css' (13 reglas), 'menu_principal.css' (26 reglas), 'index.css' (53 reglas), 'formulario_acceso.css' (27 reglas), and 'barra_estado.css'. The main pane displays the CSS code for the 'css' file, which includes font-face declarations for Roboto font at different weights and font-variants.

```

1 /* cyrillic-ext */
2 @font-face {
3   font-family: 'Roboto';
4   font-style: normal;
5   font-weight: 400;
6   src: url(https://fonts.gstatic.com/s/roboto/v32/KFOmCnqEu92Fr1Mu72xK0zY.woff2) format('woff2');
7   unicode-range: U+0460-052F, U+1C80-1C88, U+20B4, U+2DE0-2DFF, U+A640-A69F, U+FE2E-FE2F;
8 }
9 /* cyrillic */
10 @font-face {
11   font-family: 'Roboto';
12   font-style: normal;
13   font-weight: 400;
14 }

```

En la pestaña **Accesibilidad** se muestra un análisis que determina si todos los elementos de la página siguen unas mínimas reglas de accesibilidad para personas con diferentes tipos de discapacidades.

The screenshot shows the Firefox Developer Tools Accessibility tab. On the left, a tree view shows the roles of various elements on the page, such as 'generic' and 'form'. On the right, the 'Verificaciones' panel is expanded, showing sections for 'Etiquetas de texto y nombres' (with a warning about missing titles) and 'Propiedades' (listing attributes like name, role, actions, and value).

Por último, en la pestaña **Almacenamiento**, se muestran los diferentes elementos que el navegador almacena en el equipo del cliente, en relación a la página que ha cargado (sesión, cookies y caché).

The screenshot shows the Firefox Developer Tools Storage tab. The left sidebar lists storage types: 'Almacenamiento local', 'Almacenamiento de sesión', 'Almacenamiento en caché', 'Cookies', and 'Indexed DB'. The 'Cookies' section is currently selected, showing a table of stored cookies. One cookie, 'PHPSESSID', is highlighted. The table columns include Nombre, Valor, Domain, Path, Expires / Max-Age, Tamaño, HttpOnly, Secure, SameSite, and Últim.

Nombre	Valor	Domain	Path	Expires / Max-Age	Tamaño	HttpOnly	Secure	SameSite	Últim
PHPSESSID	j51d90mgfb3eil...	localhost	/	Sesión	35	false	false	None	Thu, 26 Sep 2024 18:53:54 UTC

5.2. HERRAMIENTAS DE GOOGLE CHROME

En Google Chrome también se puede acceder a las herramientas de depuración seleccionando en el menú principal **Más herramientas-> Herramientas para desarrolladores**. La pestaña de más a la izquierda se llama **Elementos** y contiene todos los elementos de la página HTML (arriba). Dentro de ella se pueden acceder a las siguientes pestañas, que se muestran en la parte de abajo: **Estilos** (con los atributos CSS de cada elemento), **Calculados** (el estilo que finalmente se aplica a cada elemento HTML), **Diseño** (si existen diseños de cuadrícula o elementos flexbox en la página), **Procesadores de eventos** (eventos JavaScript), Propiedades (de cada elemento HTML), **Puntos de interrupción DOM** (paradas que se colocan en el código JavaScript para depurar el código) y **Accesibilidad** (características que permiten que la página sea más accesible para discapacitados).

The screenshot shows the Google Chrome DevTools interface with the 'Elements' tab selected. At the top, the DOM tree is displayed, showing the structure of the page with various HTML elements like head, body, nav, and div. Below the tree, the 'Style' panel is open, showing the calculated styles for the selected element (div.headsup). It lists properties such as margin, text-align, color, padding, height, border-bottom, background-color, and display. The 'Style' tab is active, while other tabs like 'Calculated', 'Design', 'Events', and 'Accessibility' are visible but inactive. The bottom part of the screenshot shows the browser's address bar and some status icons.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="es">
  <head> ...
  </head>
  <body class="docs">
    <nav id="head-nav" class="navbar navbar-fixed-top"> ...
    <div class="headsup"> == $0
      <a href="/index.php#2024-09-26-4">PHP 8.1.30 Released!</a>
    </div>
    <nav id="trick"> ...
    <div id="goto"> ...
    <div id="breadcrumbs" class="clearfix"> ...
    <div id="layout" class="clearfix"> ...
    <!-- layout -->
    <footer> ...
    <!-- External and third party libraries. -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256- -/xUj+3OJU5yExlq6GSYGHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous">
    </script>
    <script src="/cached.php?t=1707321815&f=/js/ext/hogan-3.0.2.min.js">
    </script>
  </div>
</body>
</html>
```

html body.docs. div.headsup

Estilos Calculados Diseño Procesadores de eventos Accesibilidad >

Filtrar :hover .cls + □

```
element.style { }
}

.headsup, .headsup a {
  margin: 0 auto;
  text-align: center;
  color: #fff;
}

.headsup {
  padding: 0.25rem 0;
  height: 1.5rem;
  border-bottom: 0.125rem solid #696;
  background-color: #9c9;
  color: #fff;
}

div {
  display: block;
  width: 100px;
}
```

cached.php?...ium.css:515
cached.php?...ium.css:506
hoja de estilo de user-agent

En la pestaña de arriba, llamada **Consola**, aparecen todos los errores que se producen cuando el navegador interpreta la página (tanto errores HTML como JavaScript). En la pestaña **Fuentes** se puede inspeccionar todos los archivos de código fuente (tanto HTML como JavaScript) que se incluyen en la página. Seguidamente, en la pestaña **Red**, se muestran todos los archivos descargados en la página actual. En la pestaña **Rendimiento** aparecen estadísticas con el tiempo y la velocidad de descarga de las páginas web. Así mismo, en la pestaña **Aplicación** se muestra información sobre el almacenamiento local de información (sesiones, cookies, cache, etc.), además de los servicios en segundo plano. En la pestaña **Seguridad** se muestra información sobre la seguridad de la página, si usa el protocolo HTTPS y si los certificados son válidos.

6. ENVÍO DE CORREOS ELECTRÓNICOS A LOS USUARIOS REGISTRADOS

Desde PHP, es posible enviar correos electrónicos a los usuarios de forma automática, para ello es necesario disponer de una cuenta de correo desde donde se realizan los envíos. Si se desean enviar correos electrónicos masivos, se puede usar una cuenta de Google Gmail gratuita, que permite el envío de un mensaje con un máximo de 100 destinatarios, además de hasta 500 correos electrónicos en un mes.

Existen varias formas de enviar correos electrónicos con PHP, la forma más sencilla es a través de la extensión **PHPMailer**. Esta extensión se puede descargar como un archivo comprimido en ZIP desde el siguiente enlace y luego descomprimirlo en una carpeta del sitio web:

<https://github.com/PHPMailer/PHPMailer>

Para poder utilizar una cuenta GMail que será usada por PHPMailer para enviar los mensajes de correo, es necesario preparar previamente esta cuenta siguiendo estos pasos:

1. Acceder a la cuenta de correo en la dirección gmail.com.
2. Pulsar en el perfil del usuario que aparece en la parte superior derecha de la página y hacer clic con el ratón en el botón **Gestionar tu cuenta de Google**.
3. Acceder a la página **Seguridad** (en el menú de la izquierda).
4. Activar la **Verificación en dos pasos** de esta cuenta.
5. En la misma página de Seguridad, pulsar en el cuadro de texto de búsqueda que aparece en la parte superior de la página o también pulsando en la opción que aparece abajo del todo titulada **Buscar en la cuenta de Google**. En este cuadro de texto poner la frase “contraseñas de aplicación”. En la lista de resultados que aparezca, seleccionar la primera opción, que es **Contraseñas de aplicación**.
6. En la página que aparece, escribir un nombre cualquiera a la aplicación (por ejemplo, “**PHPMailer**”) y pulsar el botón **Crear**.
7. En ese momento, se mostrará una ventana con la contraseña generada, que será de 16 caracteres. Copiar esa contraseña, que es la que se va a utilizar en nuestra página de PHP.

Una vez que la extensión ha sido copiada al sitio y se ha preparado la cuenta desde donde se van a enviar los mensajes, en el programa PHP, hay que incluir los archivos mediante las siguientes órdenes:

```
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
use PHPMailer\PHPMailer\SMTP;
require_once ("phpmailer\\src\\Exception.php");
require_once ("phpmailer\\src\\PHPMailer.php");
require_once ("phpmailer\\src\\SMTP.php");
```

La extensión PHPMailer define la clase **PHPMailer** con los siguientes atributos públicos:

- **SMTPDebug**: establece la información que se muestra cuando se envían los mensajes. Los valores aceptados son: **SMTP::DEBUG_OFF** (sin información), **SMTP::DEBUG_CLIENT** (mensajes del cliente de correo), **SMTP::DEBUG_SERVER** (mensajes del cliente y del servidor), **SMTP::DEBUG_CONNECTION** (mensajes del servidor y de la conexión) y **SMTP::DEBUG_LOWLEVEL** (mensajes de bajo nivel, raramente se utilizan).
- **Debugoutput**: indica el formato de los mensajes de error, que puede ser: **'echo'** (texto sin formato), **'html'** (formato HTML) o **'error_log'** (formato PHP).
- **SMTPAuth**: especifica si es necesario realizar la autenticación del protocolo SMTP, el valor habitual para una cuenta de GMail es **true**.
- **SMTPSecure**: especifica el tipo de cifrado que se usa en la conexión SMTP. Valores aceptados son: **PHPMailer::ENCRYPTION_STARTTLS** (TLS, valor usado en las cuentas de GMail) o **PHPMailer::ENCRYPTION_SMTPS** (SMTPS).
- **Host**: especifica el nombre del servidor de correo electrónico. Para el servidor de Google, el valor es **'ssl://smtp.gmail.com'**.
- **Port**: especifica el número de puerto usado por el servidor de correo electrónico. El valor habitual es 25, aunque para el servidor de Google este valor es 465 (otros suelen usar el 587).

- **Username**: indica el nombre de usuario de la cuenta de correo. Para una cuenta GMail, el usuario es '`nombrecuenta@gmail.com`'.
- **Password**: indica la contraseña de la cuenta de correo. Para una cuenta de GMail, aquí hay que indicar la contraseña de aplicación que se ha generado anteriormente.
- **Subject**: especifica el título del mensaje de correo a enviar.
- **ErrorInfo**: guarda una cadena de texto con información detallada sobre el tipo de error que se ha producido en el último intento de envío de un correo electrónico.

La clase `PHPMailer` también incluye los siguientes métodos:

- `isSMTP()`: indica que se use el protocolo SMTP para el envío, es lo habitual para la mayoría de cuentas de correo, incluyendo a GMail.
- `setFrom(dirección, nombre)`: añade una dirección de correo para indicar quién envía el mensaje. Se indica la dirección de correo y, opcionalmente, un nombre. Sólo se guardará la última dirección definida.
- `addReplyTo(dirección, nombre)`: añade una dirección de correo para indicar dónde se van a enviar las respuestas al correo actual. Se indica la dirección y, opcionalmente, un nombre. Se pueden añadir varias de estas direcciones al mismo mensaje.
- `msgHTML(mensaje, ruta)`: define el mensaje a enviar en el correo electrónico, en formato HTML. Opcionalmente, si este mensaje incluye imágenes con el elemento ``, se puede especificar una ruta donde se encuentran éstas.
- `addAddress(dirección, nombre)`: añade una dirección de correo como destinatario, especificando una cadena de texto con la dirección y otra con el nombre (opcional). Se pueden añadir varias direcciones de destinatarios diferentes al mismo mensaje de correo.
- `addCC(dirección, nombre)`: añade una dirección de correo como destinatario que recibirá una copia del mismo (los demás destinatarios verán su dirección). Se especifica una cadena de texto con la dirección y otra con el nombre (opcional). Se pueden añadir varias direcciones de destinatarios diferentes al mismo mensaje.
- `addBCC(dirección, nombre)`: añade una dirección de correo como destinatario que recibirá una copia del mismo (el resto de destinatarios no verán su dirección). Se especifica una cadena de texto con la dirección y otra con el nombre (opcional). Se pueden añadir varias direcciones de destinatarios diferentes al mismo mensaje.
- `addAttachment(archivo, alias)`: añade un archivo como adjunto al mensaje, indicando una cadena de texto con su nombre y su ubicación. Opcionalmente, también se puede indicar un alias para el archivo, que será el nombre que se muestre al destinatario. Se pueden añadir varios archivos adjuntos al mismo mensaje, pero hay que conocer el tamaño máximo que es capaz de enviar la cuenta de correo utilizada.
- `send()`: devuelve `true` en caso de que el mensaje se haya enviado correctamente, `false` en caso de error.

Los métodos anteriores pueden producir una excepción de tipo `Exception` en caso de que se produzca algún problema. Así, ésta puede ser capturada perfectamente en el programa usando un bloque de sentencias `try` y `catch()`.

Se puede encontrar información más detallada sobre el uso de la clase `PHPMailer` en el siguiente enlace:

<https://phpmailer.github.io/PHPMailer/classes/PHPMailer-PHPMailer-PHPMailer.html>

7. EVALUACIÓN DEL CÓDIGO EN UNA CADENA DE TEXTO

La función `eval()`, disponible desde PHP 4, acepta como parámetro una cadena de texto y la evalúa como si fuera código a ejecutar. Por esta razón, la cadena pasada debe ser código PHP correctamente escrito. Ejemplos:

```
eval('echo "Hola, mundo<br>";');
$var=5;
echo 'Tengo '.eval('return $var;').' euros.<br>';
eval('$x="Esto es una prueba";');
echo "$x<br>";
```

El uso de la función `eval()` es muy peligroso, porque permite la ejecución de código de PHP arbitrario. Su uso está totalmente desaconsejado. Si se ha verificado cuidadosamente que no existe otra opción que usar esta función, se ha de poner especial atención en no pasar a esta función ninguna información proporcionada por el usuario sin haberla validado apropiadamente con anterioridad.

Hay que tener en cuenta varias cuestiones:

- El código pasado debe ser PHP (no HTML), aunque se puede abandonar y volver a reentrar en código PHP, usando `?>` y `<?php`. Ejemplo:

```
eval('echo "En PHP<br>";?>En HTML<br><?php echo "Más PHP<br>";');
```

- El código debe finalizar en punto y coma, y debe estar correctamente construido siguiendo las normas del lenguaje PHP.
- Una instrucción `return` dentro de la cadena a ejecutar hará que se finalice la evaluación del código inmediatamente y la función `eval()` devolverá ese valor.
- La cadena de texto se ejecutará dentro del ámbito del programa donde se llame. Esto significa que, cualquier variable definida o cambiada con `eval()` dentro de una función, sólo permanecerá dentro de ella.
- La función `eval()` devolverá `null`, a menos que la cadena de código a evaluar devuelva algún valor mediante la instrucción `return`, en cuyo caso devolverá ese valor.
- Si hay algún error en el código evaluado, se lanzará un error de tipo `E_PARSE (ParseError)` y el programa terminará completamente.
- Puesto que `eval()` es una construcción del lenguaje y no una función en sí, no puede ser llamada usando funciones variables.

Por último, a modo de anécdota, se incluye una frase célebre de Rasmus Lerdorf, creador de PHP, que habla del uso de la función `eval()`, de una forma bastante elocuente:

"Si eval() es la respuesta, es casi seguro que estás haciendo la pregunta equivocada".