

Desarrollo de aplicaciones web con PHP

Desarrollo Web en Entorno Servidor

1 de diciembre de 2022

CIFP Carlos III - Cartagena

Santiago Francisco San Pablo Raposo

2º curso DAW

Contenido

**No se encontraron entradas de tabla de contenido.**

# Índice de ilustraciones.

**No se encuentran elementos de tabla de ilustraciones.**

# Índice de tablas.

**No se encuentran elementos de tabla de ilustraciones.**

Desarrollo de aplicaciones web con PHP.

# 1.- Autentificación de usuarios y control de acceso.

Muchas veces es importante verificar la identidad de los dos extremos de una comunicación. En el caso de una comunicación web, existen métodos para identificar tanto al servidor en el que se aloja el sitio web, como al usuario.

Los sitios web que necesitan emplear identificación del servidor, como las tiendas o los bancos, utilizan el protocolo HTTPS. HTTPS utiliza métodos de cifrado para crear un canal seguro entre el navegador y el servidor, de tal forma que no se pueda interceptar la información.

**Para identificar a los usuarios que visitan un sitio web**, se pueden utilizar **distintos métodos:**

* Como el DNI digital,
* certificados digitales de usuario,
* pero el más extendido es solicitar al usuario cierta información que solo él conoce: **la combinación de un nombre de usuario y una contraseña**.

Es **en este tipo de aplicaciones en las que se accede a BB.DD, donde resulta interesante e importante implantar algún mecanismo de control de acceso** que obligue al usuario a identificarse. Una vez identificado, se puede limitar el uso que puede hacer de la información.

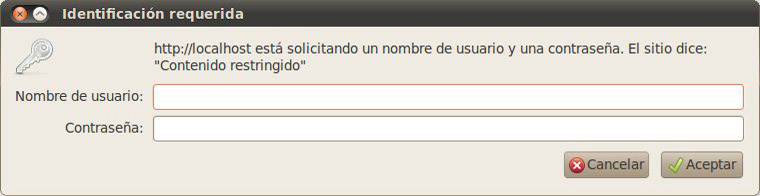
**Debes distinguir la autentificación de los usuarios y el control de acceso, de la utilización de mecanismos para asegurar las comunicaciones (HTTPS)** entre el usuario del navegador y el servidor web. Aunque ambos aspectos suelen ir unidos, son independientes.

En la unidad, todo se desarrollará con HTTP, ya que HTTPS no es el objetivo de este módulo.

## 1.1.- Mecanismos de autentificación (I).

El protocolo HTTP ofrece un método sencillo para autentificar a los usuarios:

1. **El servidor web provee de un método para identificar a los usuarios**, y mantiene una lista de control de acceso (ACL) que define los recursos que tiene restringidos.



1. **La información de autentificación del usuario se envía al servidor**, que la verifica y decide si permite o no el acceso al recurso solicitado.
2. **Ante un error de autentificación**, sale “Acceso no autorizado” (Código 401).

En Apache, el comando [**htpasswd**](https://httpd.apache.org/docs/2.0/es/howto/auth.html), permite almacenar en un fichero una lista de usuarios y sus respectivas contraseñas. La información relativa a las contraseñas se almacena cifrada; aun así, es conveniente crear este fichero en un lugar no accesible por los usuarios del servidor web.

* **Para crear usuarios**: por ejemplo: sudo htpasswd –c users dwes.
  + Una vez introducido el comando, te pedirá que le pongas una contraseña al usuario.
  + La opción -c reemplaza el archivo (es decir, machaca al anterior, si lo hubiera). Solo debe usarse para crear el archivo (es decir, la primera vez que se usa).
* **Para indicarle al servidor Apache qué recursos tienen acceso restringido**, una opción es crear un fichero **.htaccess** en el directorio en que se encuentren, con las siguientes directivas:

AuthName "Contenido restringido"

AuthType Basic

AuthUserFile /etc/apache2/users

require valid-user

**Significado de cada una de las directivas anteriores**:

|  |  |
| --- | --- |
| Directiva | Signfiicado |
| AuthName | Nombre de dominio que se usará en la autentificación. |
| Sirve para que, si el cliente se autentifica correctamente, esa misma información de autentificación se utilizará automáticamente en el resto de las páginas del mismo dominio. |
| AuthType | Método de autentificación que se usará. Además del método **Basic**, Apache también permite utilizar el método Digest. |
| AuthUserFile | Ruta al **archivo de credenciales** que has creado con htpasswd. |
| Require | Permite indicar que sólo puedan acceder algunos usuarios o grupos de usuarios concretos. |
| Si indicamos "valid-user", podrán acceder todos los usuarios que se autentifiquen correctamente |

Además, tendrás que asegurarte de que en la configuración de Apache se utiliza la directiva [**AllowOverride**](https://httpd.apache.org/docs/2.0/es/mod/core.html#allowoverride) para que se aplique correctamente la configuración que figura en los archivos **.htaccess**.

Imagen que contiene Texto

Descripción generada automáticamente

### 1.1.1.- Recuperar datos de acceso en PHP.

**Valores del array $\_SERVER relacionados con la autentificación HTTP**:

|  |  |
| --- | --- |
| Valor | Contenido |
| $\_SERVER['PHP\_AUTH\_USER'] | Nombre de usuario |
| $\_SERVER['PHP\_AUTH\_PW'] | Contraseña |
| $\_SERVER['AUTH\_TYPE'] | Método HTTP usado para autentificar. Puede ser Basic o Digest |

Además, en PHP **puedes usar la función header para forzar a que el servidor envíe un error de "Acceso no autorizado" (código 401)**. **De esta forma no es necesario utilizar ficheros .htaccess** para indicarle a Apache qué recursos están restringidos.

En su lugar, puedes añadir las siguientes líneas en tus páginas PHP:

<?php

if (!isset($\_SERVER['PHP\_AUTH\_USER'])) {

header('WWW-Authenticate: Basic Realm="Contenido restringido"');

header('HTTP/1.0 401 Unauthorized');

echo "Usuario no reconocido!";

exit;

}

?>

**Debes conocer**: la función **header** envía encabezados HTTP, pero debe utilizarse antes de que se muestre nada por pantalla. En caso contrario, obtendrás un error. [PHP: header - Manual](https://www.php.net/manual/es/function.header.php)

**Con el código anterior, la página envía un error 401, lo que fuerza al navegador a solicitar las credenciales de acceso** (nombre de usuario y contraseña). Si se introducen, se ejecuta el resto de la página y se muestra su contenido.

[**Ejercicio resuelto**](https://aula21-my.sharepoint.com/personal/5634224_alu365_murciaeduca_es/Documents/Documentos/FP/DAW/Curso%202/Desarrollo%20web%20en%20entorno%20servidor/Evaluacion%201/Tema%204/Recursos/DWES04_CONT_R02f_Funcion_header.pdf): modifica la [página anterior](https://aula21-my.sharepoint.com/personal/5634224_alu365_murciaeduca_es/Documents/Documentos/FP/DAW/Curso%202/Desarrollo%20web%20en%20entorno%20servidor/Evaluacion%201/Tema%204/Recursos/DWES04_CONT_R02c_Autentificacion_HTTP.pdf) utilizando la función header para que solicite las credenciales al usuario.

## 1.2.- Incorporación de métodos de autentificación a una aplicación web.

**Si utilizas la función** **header** para forzar al navegador a solicitar credenciales HTTP, el usuario introducirá un nombre y una contraseña. Pero **el servidor no verificará esta información; deberás ser tú quien provea un método para comprobar que las credenciales que ha introducido el usuario son correctas**.

**Método más simple**:

<?php

if ($\_SERVER['PHP\_AUTH\_USER']!='dwes' || $\_SERVER['PHP\_AUTH\_PW']!='abc123.') {

header('WWW-Authenticate: Basic Realm="Contenido restringido"');

header('HTTP/1.0 401 Unauthorized');

echo "Usuario no reconocido!";

exit;

}

?>

Aunque no es inseguro (ya que el código PHP no es visible por el usuario), **esto hará menos portable tu código**, ya que **si necesitas modificar el nombre de usuario o la contraseña, tendrás que hacer modificaciones en el código**. Además, no podrás permitir al usuario introducir su propia contraseña.

**Mejor solución**: utilizar un almacenamiento externo para los nombres de usuario y sus contraseñas. Para esto podrías emplear un fichero de texto, o mejor: **una base de datos**.

La información de autentificación podrá estar aislada en su propia base de datos, o compartir espacio de almacenamiento con los datos que utilice tu aplicación web.

**Si quieres almacenar la información de los usuarios en la base de datos "dwes"**, tienes que crear una nueva tabla en su estructura. Para ello, revisa y ejecuta estas sentencias SQL.

-- Seleccionamos la base de datos

USE dwes;

-- Creamos la tabla

CREATE TABLE usuarios (

usuario VARCHAR(20) NOT NULL PRIMARY KEY,

contrasena VARCHAR(32) NOT NULL

) ENGINE = INNODB;

-- Creamos el usuario dwes

INSERT INTO usuarios (usuario, contrasena) VALUES

('dwes', 'e8dc8ccd5e5f9e3a54f07350ce8a2d3d');

Como se puede ver, para no **almacenar las contraseñas** en texto plano, es **mejor hacerlo en formato encriptado**. En PHP puedes usar [**la función md5**](https://www.php.net/manual/es/function.md5.php), que te permite calcular el hash MD5 de una cadena de texto, para que vaya encriptado.

[**Ejercicio resuelto**](../Recursos/DWES04_CONT_R03c_Solucion_ejercicio.pdf): Utiliza la extensión MySQLi para modificar el ejercicio anterior, de tal forma que las credenciales del usuario se comprueben con la información de la nueva tabla "usuarios" creada en la base de datos "dwes". Si no existe el usuario, o la contraseña es incorrecta, vuelve a pedir las credenciales al usuario.

Texto

Descripción generada automáticamente

# 2.- Cookies.

**Una cookie es un fichero de texto que un sitio web guarda en el entorno del usuario del navegador**. Su uso más típico es el almacenamiento de las preferencias del usuario (por ejemplo, el idioma en que se deben mostrar las páginas), para que no tenga que volver a indicarlas la próxima vez que visite el sitio.

**Recomendación**: existen extensiones para los navegadores para consultar y editar las cookies de los sitios web que visitas.

**Para almacenar una cookie con PHP**: función [setcookie](https://www.php.net/manual/es/function.setcookie.php) (nombreCookie[, valor, arrayOpciones(caducidad, ruta, dominio, seguridad, httponly)]) : bool

Como otros encabezados, **cookies deben ser enviadas *antes* de cualquier salida en el script** (este es un protocolo de restricción).

**Por ejemplo**:

setcookie("nombre\_usuario", $\_SERVER['PHP\_AUTH\_USER'], time()+3600);

Los dos primeros parámetros son el nombre de la cookie y su valor. El tercero es la fecha de caducidad de la misma (una hora desde el momento en que se ejecute). En caso de no figurar este parámetro, la cookie se eliminará cuando se cierre el navegador.

Ten en cuenta que **también se pueden aplicar restricciones a las páginas del sitio que pueden acceder a una cookie en función de la ruta**.

**Recuperar datos de las cookies en PHP**: cuando accedes a un sitio web, el navegador le envía de forma automática todo el contenido de las cookies que almacene relativas a ese sitio en concreto. Desde PHP puedes acceder a esta información por medio del array **$\_COOKIE**.

**Siempre que utilices cookies en una aplicación web, debes tener en cuenta que en última instancia su disponibilidad está controlada por el cliente**. **Por ejemplo**, algunos usuarios deshabilitan las cookies en el navegador porque piensan que la información que almacenan puede suponer un potencial problema de seguridad.

**Recomendación**: para eliminar una cookie antes de su fecha de expiración: basta con usar nuevamente setcookie estableciendo una fecha de expiración más temprana para que caduque antes.

[**Ejercicio resuelto**](../Recursos/DWES04_CONT_R04b_Solucion_ejercicio.pdf): Sobre el mismo ejercicio anterior, almacena en una cookie el último instante en que el usuario visitó la página. Si es su primera visita, muestra un mensaje de bienvenida. En caso contrario, muestra la fecha y hora de su anterior visita.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# 3.- Manejo de sesiones.

Aunque las **cookies** nos permiten almacenar la información de las credenciales del usuario en el equipo del usuario, podemos tener **problemas** con el **nº de ellas que permite el navegador, o su tamaño máximo**. Para solventar esto, **están las sesiones**.

**Sesión**: conjunto de información relativa a un usuario concreto. Esta información puede ser tan simple como el nombre del propio usuario, o más compleja, como los artículos que ha depositado en la cesta de compra de una tienda online.

**Para distinguir una sesión de otra se usan los identificadores de sesión (SID)**. Un SID es un atributo que se asigna a cada uno de los visitantes de un sitio web y lo identifica. De esta forma, si el servidor web conoce el SID de un usuario, puede relacionarlo con toda la información que posee sobre él, que se mantiene en la sesión del usuario.

**Esa información se almacena** en el servidor web, **generalmente en ficheros**, **aunque también se pueden** utilizar otros mecanismos de almacenamiento como **bases de datos**.

**¿Y dónde se almacena ese SID?**

* Un método es utilizando **cookies**, como hemos visto ya.
* **Propagando el SID en un parámetro de la URL**. El SID se añade como una parte más de la URL, de la forma:
  + <http://www.misitioweb.com/tienda/listado.php&PHPSESSID=34534fg4ffg34ty>

**Ninguna de las dos maneras es perfecta**.

* Ya sabes los problemas que tiene la utilización de cookies. Pese a ello, es el mejor método y el más utilizado.
* **Propagar el SID como parte de la URL conlleva mayores desventajas**, como la imposibilidad de mantener el SID entre distintas sesiones, o el hecho de que compartir la URL con otra persona implica compartir también el identificador de sesión.

La buena noticia es que el proceso de manejo de sesiones en PHP está automatizado en gran medida. No hay que programar un procedimiento para comprobar si existe un SID previo y en ese caso, cargar sus datos, ni usar la funcion **setcookie**. **Todo esto lo hace PHP automáticamente por ti**.

**A la información que se almacena en la sesión de un usuario también se le conoce como cookies del lado del servidor** (server side cookies). **Aunque esta información no viaja entre el cliente y el servidor, sí lo hace el SID** (como parte de la URL o en un encabezado HTTP si se guarda en una cookie).

**La manera más segura de utilizar sesiones es almacenando los SID en cookies y utilizar HTTPS**para encriptar la información que se transmite entre el servidor web y el cliente.

## 3.1.- Configuración.

**Antes de utilizar sesiones en tu sitio web, debes configurar correctamente PHP** utilizando las siguientes directivas **en el fichero php.ini** según corresponda.

|  |  |
| --- | --- |
| Directiva | Signficiado |
| session.use\_cookies | * 1: Indica si se deben usar cookies * 2: o propagación en la URL para almacenar el SID. |
| session. use\_only\_cookies | Se debe activar (1) **cuando utilizas cookies para almacenar los SID, y además no quieres que se reconozcan los SID que se puedan pasar como parte de la URL** (este método se puede usar para usurpar el identificador de otro usuario). |
| session.save\_handler | Se utiliza para indicar a PHP cómo debe almacenar los datos de la sesión del usuario. **4 opciones**:   * en ficheros (**files**): por defecto si no se especifica nada. * en memoria (**mm**) * en una base de datos SQLite (**sqlite**) * utilizando para ello funciones que debe definir el programador (**user**). |
| session.name | **Nombre de la cookie** que se utilizará **para guardar el SID**. Su valor por defecto es PHPSESSID. |
| session.auto\_start | * **0**: no se inician automáticamente. En este caso, deberás usar la función session\_start para gestionar el inicio de las sesiones. * **1**: se inician automáticamente. |
| session. cookie\_lifetime | * Si utilizas la URL para propagar el SID, éste se perderá cuando cierres tu navegador. **Sin embargo, si utilizas cookies, el SID se mantendrá mientras no se destruya la cookie**. * **0 (por defecto)**: las cookies se destruyen cuando se cierra el navegador. * **Otros valores**: si quieres que se mantenga el SID durante más tiempo, debes indicar en esta directiva ese tiempo en segundos. |
| session. gc\_maxlifetime | Indica el **tiempo en segundos que se debe mantener activa la sesión**, aunque no haya ninguna actividad por parte del usuario. Su valor por defecto es 1440 (24 minutos). |

**Como consultar la configuración actual de estas directivas**: mediante la función phpinfo();

**Para saber más**: [sobre estas y otras directivas](https://www.php.net/manual/es/session.configuration.php) que permiten configurar el manejo de sesiones.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

## 3.2.- Inicio y fin de una sesión.

**Si está activada la directiva session.auto\_start** en la configuración de PHP, la sesión comenzará automáticamente en cuanto un usuario se conecte a tu sitio web.

**En caso de que ese usuario ya haya abierto una sesión con anterioridad, y esta no se haya eliminado**, en lugar de abrir una nueva sesión **se reanudará la anterior**. **Para ello se utilizará el SID anterior, que estará almacenado en una cookie** (recuerda que, si usas propagación del SID, no podrás restaurar sesiones anteriores; el SID figura en la URL y se pierde cuando cierras el navegador).

**Si no**: debes usar la función **session\_start** de PHP, que devolverá true si se inicia correctamente la sesión, y false si no.

|  |
| --- |
| Prestar atención |
| Como el inicio de sesión requiere utilizar cookies, y éstas se transmiten en los encabezados HTTP, debes tener en cuenta que para poder iniciar una sesión utilizando **session\_start**, tendrás que hacer las llamadas a esta función antes de que la página web muestre información en el navegador.  Además, todas las páginas que necesiten utilizar la información almacenada en la sesión, deberán ejecutar la función **session\_start**. |

Mientras la sesión permanece abierta, puedes utilizar la variable superglobal **$\_SESSION** para añadir información a la sesión del usuario. **Por ejemplo**: un contador de visitas del usuario.

<?php

// Iniciamos la sesión o recuperamos la anterior sesión existente

session\_start();

// Comprobamos si la variable ya existe

if (isset($\_SESSION['visitas']))

$\_SESSION['visitas']++;

else

$\_SESSION['visitas'] = 0;

?>

Si en lugar del número de visitas, quisieras almacenar el instante en que se produce cada una

<?php

// Iniciamos la sesión o recuperamos la anterior sesión existente

session\_start();

// En cada visita añadimos un valor al array "visitas"

$\_SESSION['visitas'][] = mktime();

?>

**Dos funciones para eliminar la información de sesión almacenada**:

En ocasiones puede ser necesario cerrarla de forma manual en un determinado momento. **Por ejemplo**: deberás darle al usuario del sitio web la posibilidad de cerrar la sesión cuando lo crea conveniente.

* **session\_unset**: **elimina** las **variables** almacenadas en la sesión actual, pero no elimina la información de la sesión del dispositivo de almacenamiento usado.
* **session\_destroy**: **elimina** completamente la información de **la sesión** del dispositivo de almacenamiento.

[**Ejercicio resuelto**](https://aula21-my.sharepoint.com/personal/5634224_alu365_murciaeduca_es/Documents/Documentos/FP/DAW/Curso%202/Desarrollo%20web%20en%20entorno%20servidor/Evaluacion%201/Tema%204/Recursos/DWES04_CONT_R07b_Solucion_ejercicio.pdf): Crea una página similar a la anterior, almacenando en la sesión de usuario los instantes de todas sus últimas visitas. Si es su primera visita, muestra un mensaje de bienvenida. En caso contrario, muestra la fecha y hora de todas sus visitas anteriores. Añade un botón a la página que permita borrar el registro de visitas.

Utiliza también una variable de sesión para comprobar si el usuario se ha autentificado correctamente. De esta forma no hará falta comprobar las credenciales con la base de datos constantemente.

Imagen que contiene Aplicación

Descripción generada automáticamente

## 3.3.- Gestión de la información de la sesión.

Esto vamos a verlo con un ejemplo: utilizarás la base de datos "dwes", creada anteriormente, para crear un prototipo de una tienda web dedicada a la venta de productos de informática.

* **Login (login.php)**. Su función es autentificar al usuario de la aplicación web. Todos los usuarios pasan por esta página antes de acceder al resto.
* **Listado de productos (productos.php)**. Presenta un listado de los productos de la tienda
* **Cesta de compra (cesta.php)**. Muestra un resumen de los productos escogidos por el usuario.
* **Pagar (pagar.php)**. Una vez confirmada la compra, la última página debería ser la que permitiera al usuario escoger el método de pago y la forma de envío.
  + En este ejemplo no la vas a implementar como tal. Simplemente mostrará un mensaje de tipo "Gracias por su compra" y ofrecerá un enlace para comenzar una nueva compra
* **Logoff (logoff.php)**. Esta página desconecta al usuario de la aplicación y redirige al usuario de forma automática a la pantalla de autentificación.

Usaremos esta [hoja de estilos](https://aula21-my.sharepoint.com/personal/5634224_alu365_murciaeduca_es/Documents/Documentos/FP/DAW/Curso%202/Desarrollo%20web%20en%20entorno%20servidor/Evaluacion%201/Tema%204/Recursos/DWES04_CONT_R08b_tienda.css) y esta imagen para la [cesta](https://aula21-my.sharepoint.com/personal/5634224_alu365_murciaeduca_es/Documents/Documentos/FP/DAW/Curso%202/Desarrollo%20web%20en%20entorno%20servidor/Evaluacion%201/Tema%204/Recursos/DWES04_CONT_R08c_cesta.jpg).png (carrito).

**Haremos algunas simplificaciones** (además de no proporcionar métodos de pago reales):

* **No se podrán comprar varias unidades** del mismo producto.
* **Una vez añadido un producto a la cesta de compra, no se podrá retirar de la misma**. La única posibilidad será vaciar toda la cesta.
* **No se mostrarán imágenes de los productos**, ni será posible ver el total de la compra hasta que ésta haya finalizado.
* **Se muestran todos los productos en una única página**. Sería preferible filtrarlos por familia y mostrarlos en varias páginas, limitando a 10 o 20 productos el número máximo de cada página.

### 3.3.1.- login.php

**Código HTML**:

<div id='login'>

<form action='login.php' method='post'>

<fieldset >

<legend>Login</legend>

<div><span class='error'><?php echo $error; ?></span></div>

<div class='campo'>

<label for='usuario' >Usuario:</label><br/>

<input type='text' name='usuario' id='usuario' maxlength="50" /><br/>

</div>

<div class='campo'>

<label for='password' >Contraseña:</label><br/>

<input type='password' name='password' id='password' maxlength="50" /><br/>

</div>

<div class='campo'>

<input type='submit' name='enviar' value='Enviar' />

</div>

</fieldset>

</form>

</div>

**Código PHP**:

if (isset($\_POST['enviar'])) {

$usuario = $\_POST['usuario'];

$password = $\_POST['password'];

if (empty($usuario) || empty($password)) // Comprobar credenciales

$error = "Debes introducir un nombre de usuario y una contraseña";

else { // Conectarse a la BB.DD

try {

$opc = array(PDO::MYSQL\_ATTR\_INIT\_COMMAND => "SET NAMES utf8");

$dsn = "mysql:host=localhost;dbname=dwes";

$dwes = new PDO($dsn, "dwes", "abc123.", $opc);

}

catch (PDOException $e) {

die("Error: " . $e->getMessage());

}

// Comprobar las credenciales

$sql = "SELECT usuario FROM usuarios WHERE usuario='$usuario' " .

"AND contrasena='" . md5($password) . "'";

if($resultado = $dwes->query($sql)) {

$fila = $resultado->fetch();

if ($fila != null) {

// Inicia sesión y almacena el nombre de usuario

session\_start();

$\_SESSION['usuario']=$usuario;

// Redirige a la página del listado de productos.

header("Location: productos.php");

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

### 3.3.2.- productos.php

La página consta de:

* **Encabezado**: contiene el título de la página.
* **Productos**: es la lista de productos. Se crea un formulario por cada producto, con un botón “Añadir” que envía a esta misma página los datos código, nombre y precio del producto. Cuando se abre la página, se comprueba si se ha enviado este formulario. Y si se ha enviado, añade un elemento al array $\_SESSION[‘cesta’] con los datos del nuevo producto.

// Comprobamos si se ha enviado el formulario de añadir

if (isset($\_POST['enviar'])) {

// Creamos un array con los datos del nuevo producto

$producto['nombre'] = $\_POST['nombre'];

$producto['precio'] = $\_POST['precio'];

// y lo añadimos

$\_SESSION['cesta'][$\_POST['producto']] = $producto;

}

El array **$\_SESSION['cesta']** es la variable de sesión en la que guardaremos los datos de todos los productos que va a comprar el usuario

* **Cesta**: muestra el código de los productos que se van añadiendo a la cesta.

// Si la cesta está vacía, mostramos un mensaje

$cesta\_vacia = true;

if (count($\_SESSION['cesta'])==0) {

print "<p>Cesta vacía</p>";

}

// Si no está vacía, mostrar su contenido

else {

foreach ($\_SESSION['cesta'] as $codigo => $producto)

print "<p>$codigo</p>";

$cesta\_vacia = false;

}

Contiene dos formularios: uno para vaciar la cesta (botón “Vaciar cesta”, y otro para realizar la compra, que dirige a la página **cesta.php**.

**Para vaciar la vesta**:

// Comprobamos si se ha enviado el formulario de vaciar la cesta

if (isset($\_POST['vaciar'])) {

unset($\_SESSION['cesta']);

}

* **Pie**: Contiene un botón para desconectar al usuario actual. Llama a la página **logoff.php**, que borra la sesión actual
* **Comprobación de la sesión**: como viene siendo habitual, en todas las páginas que requieran autentificación, requerirán de esta comprobación.

<?php

// Recuperamos la información de la sesión

session\_start();

// Y comprobamos que el usuario se haya autentificado

if (!isset($\_SESSION['usuario'])) {

die("Error - debe <a href='login.php'>identificarse</a>.<br />");

}

?>

### 3.3.3.- cesta.php, pagar.php y logoff.php

Si el usuario pulsa el botón “Comprar” de la página del listado de productos, se le redirige a **cesta.php** (muestra un resumen de los productos que ha seleccionado junto al importe total de los mismos).



El formulario que contiene al botón “Pagar” redirige a la página **pagar.php**, que, en nuestro caso, lo único que hace es eliminar la sesión del usuario. Y el que contiene al botón “Desconectar usuario”, redirige a la página **logoff.php**, que cierra la sesión del usuario.

**Los datos que figuran en la página se obtienen todos de la información almacenada en la sesión del usuario** sin necesidad de establecer conexiones con la base de datos.

**Para mostrar el listado de los productos seleccionados es el siguiente**:

<?php

$total = 0;

foreach($\_SESSION['cesta'] as $codigo => $producto) {

echo "<p><span class='codigo'>$codigo</span>";

echo "<span class='nombre'>${producto['nombre']}</span>";

echo "<span class='precio'>${producto['precio']}</span></p>";

$total += $producto['precio'];

}

?>

<hr />

<p><span class='pagar'>Precio total: <?php print $total; ?> €</span></p>

Recuerda que al principio de esta página, **también será necesario verificar la variable de sesión** $\_SESSION['usuario'] para comprobar que el usuario se ha autentificado correctamente

**Para cerrar sesión**: tenemos el archivo logoff.php con el siguiente código:

<?php

// Recuperamos la información de la sesión

session\_start();

// Y la eliminamos

session\_unset();

header("Location: login.php");

?>

Interfaz de usuario gráfica

Descripción generada automáticamente

# 4.- Herramientas para depuración de código.

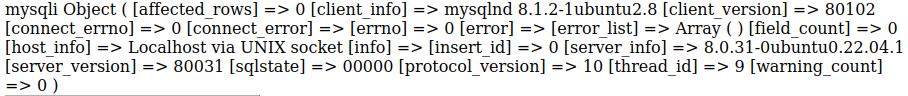
Para descubrir que es lo que está fallando, tienes varias opciones:

* [**var\_dump**](https://www.php.net/manual/es/function.var-dump.php): imprime lo que contiene una variable/array. **Ejemplo de presentación**:

Texto

Descripción generada automáticamente

* **print\_r**: igual, pero la presentación de la información es esta:



**Estos métodos de depuración son muy tediosos** y requieren hacer cambios constantes en el código de la aplicación. Además, una vez depurado el código, **se nos puede olvidar quitarlos**.

**Por ello, se utilizan herramientas de depuración** que se integran con el IDE. De las herramientas disponibles que posibilitan la depuración en lenguaje PHP, en esta unidad aprenderás a configurar y utilizar la extensión **Xdebug**.

Es una extensión de código libre, bajo licencia propia (The Xdebug License, basada en la licencia de PHP) que se integra perfectamente con el IDE que estás usando, NetBeans.

Si la depuración está activada, Xdebug controla la ejecución de los guiones en PHP. Puede pausar, reanudar y detener los programas en cualquier momento. Cuando el programa está pausado, Xdebug puede obtener información del estado de ejecución, incluyendo el valor de las variables (puede incluso cambiar su valor).

**Xdebug** **es un servidor** que recibe instrucciones de un cliente, que en nuestro caso será NetBeans. **De esta forma, no es necesario que el entorno de desarrollo esté en la misma máquina que el servidor PHP con Xdebug**.

## 4.1.- Instalación de herramientas para depuración.

Aunque existen diversas formas para realizar la instalación de Xdebug. Probablemente la más sencilla es a través del repositorio de extensiones PECL, del que ya hablamos en la unidad 2.

1. Necesitas de que PECL está disponible en tu instalación de PHP. Para instalarlo desde los repositorios de Ubuntu, ejecuta desde una consola:

sudo apt-get update

sudo apt-get install php-pear

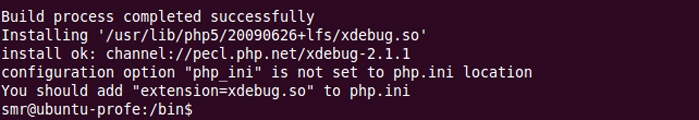
1. Además, para poder compilar las extensiones que descargues con PECL, debes instalar el paquete con las herramientas de desarrollo php5-dev. Incluye dos programas necesarios para instalar Xdebug: **phpize** y **php-config**.

sudo apt-get install php5-dev

1. Instalamos Xdebug desde el repositorio PECL:

sudo pecl install xdebug

1. Al finalizar la instalación, obtendremos un mensaje como este:



1. Para solucionarlo, nos vamos al fichero php.ini y añadimos las siguientes líneas al final del fichero:

[xdebug]

zend\_extension="/usr/lib/php5/20090626+lfs/xdebug.so"

; Opciones de configuración

xdebug.remote\_enable=on

xdebug.remote\_handler=dbgp

xdebug.remote\_host=localhost

xdebug.remote\_port=9000

Ajusta la ruta anterior al fichero xdebug.so para que sea la misma que has obtenido en tu sistema tras instalar la extensión.

1. Recuerda **reiniciar el servidor Apache**:

sudo /etc/init.d/apache2 restart

**Para saber más**: Si necesitas instalar Xdebug en sistemas Windows, puedes encontrar información al respecto en Internet. En la página web de la extensión Xdebug tienes información en lenguaje inglés. [Xdebug: Documentation » Installation](https://xdebug.org/docs/install)

## 4.2.- Depuración de código en PHP.

**Características que destacan de Xdebug**:

* Creación de registros con las llamadas a funciones que se produzcan, incluyendo parámetros y valores devueltos.
* Creación de registros de optimización, que permitan analizar el rendimiento de los programas.
* **Depuración remota**.

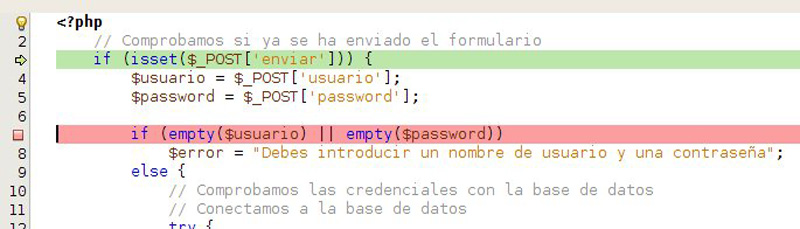
**Nos centraremos en la depuración remota**:

1. Al iniciar la depuración, se nos detendrá en la primera línea (salvo que configuremos el comportamiento por defecto para que no lo haga).

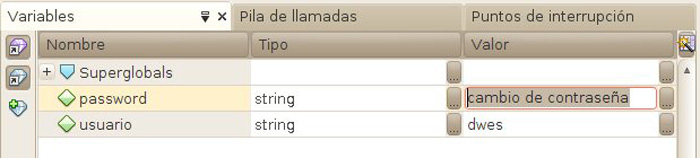
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. Podremos añadir puntos de interrupción en una línea de código.



1. También puedes modificar el valor de las variables editando directamente sobre la zona en que se muestran:



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente