

## Tarea online

### 1.- Descripción de la tarea

*1.1.- Ejercicio 1: jugando con los tipos mime*

*1.2.- Ejercicio 2: un texto brillante*

*1.3.- Ejercicio 3: solo personal autorizado*

*1.4.- Ejercicio 4: vamos paso a paso*

### 2.- Información de interés

### 3.- Evaluación de la tarea

### Anexo. Licencia de recursos

# Tarea online

**Título de la tarea:** Desarrollo de aplicaciones web con PHP.

**Unidad:** 3.

**Ciclo formativo y módulo:** DAW, Desarrollo Web en Entorno Servidor.

**Curso académico:** 2020/21

## ¿Qué contenidos o resultados de aprendizaje trabajaremos?

En esta tarea se trabajarán aspectos tales como:

- ✓ Manipulación de cabeceras HTTP.
- ✓ Uso de cookies.
- ✓ Uso de sesiones.
- ✓ Almacenamiento de información en archivos.

Los aspectos anteriores están centrados en la consecución del resultado de aprendizaje 4: "Desarrolla aplicaciones Web embebidas en lenguajes de marcas analizando e incorporando funcionalidades según especificaciones. aunque también contribuye indirectamente a la consecución del resultado de aprendizaje 1, 2 y 3.

## Tarea online

### 1.- Descripción de la tarea

1.1.- Ejercicio 1: jugando con los tipos mime

1.2.- Ejercicio 2: un texto brillante

1.3.- Ejercicio 3: solo personal autorizado

1.4.- Ejercicio 4: vamos paso a paso

### 2.- Información de interés

### 3.- Evaluación de la tarea

### Anexo. Licencia de recursos

## 1.1.- Ejercicio 1: jugando con los tipos mime

### Debes conocer

A través del protocolo HTTP el servidor web envía HTML al navegador web, HTML que después el navegador web renderiza y transforma en algo visible para el usuario. Como ya sabrás, HTML permite crear documentos que pueden incluir referencias a otros documentos o archivos necesarios para mostrar la página web de forma adecuada. Se trata de archivos tales como: imágenes, vídeos, sonidos, hojas de estilo, etc.; en definitiva, archivos adicionales que conforman el aspecto final de la página web. Pues bien, como ya sabrás, esos archivos adicionales se envían también al navegador web usando el protocolo HTTP.

Esto quiere decir que a través del protocolo HTTP no solo se envían archivos HTML al navegador web, sino que también se envían otros tipos de datos (imágenes, vídeos, etc.). Cuando el navegador web inicia una petición HTTP (HTTP REQUEST), en la respuesta HTTP generada por el servidor web (HTTP RESPONSE) se indica el tipo de dato que se envía al cliente. Fíjate en la siguiente petición web (y su respuesta), donde se solicita una imagen:

Petición del cliente web (HTTP REQUEST)	Respuesta del servidor web (HTTP RESP)
<pre>GET /imgs/image.jpg HTTP/1.1 Host: www.examplehost.reallyidontexists User-Agent: Mozilla/5.0 (...) Gecko/20100101 Firefox/83.0 Accept: image/webp, */* Accept-Language: es-ES, es; q=0.8, en-US; q=0.5, en; q=0.3 Accept-Encoding: gzip, deflate, br Connection: keep-alive Referer: ... Cookie: ...</pre>	<pre>HTTP/1.1 200 OK Date: Thu, 26 Nov 2020 12:48:53 GMT Server: Apache/2.4.38 (Debian) Expires: Mon, 25 Jan 2021 12:48:53 GMT Cache-Control: public, max-age=5184000, no-t Pragma: Content-Disposition: inline; filename="image Last-Modified: Fri, 11 Sep 2020 10:00:45 GMT Etag: "c6d8be40f6024b4c27b6b77f1049631a95a53 Accept-Ranges: bytes Content-Length: 176328 Content-Type: image/jpeg Connection: Keep-Alive</pre>

Ahora necesito que te fijas no tanto en la petición del cliente web, sino en la respuesta del servidor. Fíjate que se incluye información sobre el tipo de archivo que se está enviando al navegador:

- ✓ **Content-Length: 176328** . Cantidad de bytes enviados al navegador (corresponde con el peso de la imagen o del recurso enviado).
- ✓ **Content-Type: image/jpeg** . Tipo MIME del dato enviado.

Esos trozos de texto corresponden a dos cabeceras que permiten indicar el tipo de contenido enviado al cliente, y se pueden manipular desde PHP a través de la función `header`.

[Referencia de la función header](#)

En este ejercicio harás un script PHP que sirva diferente tipo de contenido en función de un parámetro `GET`:

- ✓ `http://localhost/dwes03/ejercicio1.php?n=0` se servirá un contenido tipo HTML (usar el tipo mime `'text/html'`).
- ✓ `http://localhost/dwes03/ejercicio1.php?n=1` se servirá un contenido tipo PDF (usar el tipo mime `'application/pdf'`).
- ✓ `http://localhost/dwes03/ejercicio1.php?n=2` se servirá un contenido tipo JPG (usar el tipo mime `'image/jpeg'`).
- ✓ `http://localhost/dwes03/ejercicio1.php?n=3` se servirá un contenido tipo PNG (usar el tipo mime `'image/png'`).
- ✓ `http://localhost/dwes03/ejercicio1.php?n=4` se servirá un contenido tipo MP4 que contiene vídeo (usar el tipo mime `'video/mp4'`).
- ✓ `http://localhost/dwes03/ejercicio1.php?n=5` se servirá un contenido tipo MP4 que contiene solo audio (usar el tipo mime `'audio/mp4'`).

**Importante:** en caso de que no se especifique el parámetro `n` o este no sea correcto, deberá mostrarse el texto en HTML (el mismo que cuando `n` es 0).

Para ello debes usar las siguientes funciones:

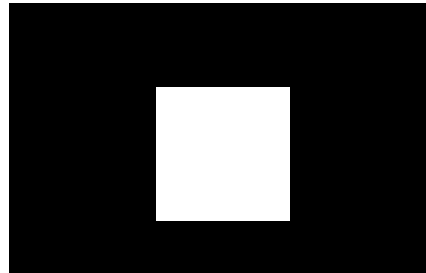
- ✓ Función [header](#): para modificar las cabeceras de la respuesta HTTP, estableciendo el tamaño del archivo y el tipo MIME (ten en cuenta que esta función debe usarse antes de que el script php genere ningún texto).
- ✓ Función [filesize](#): para obtener el tamaño en bytes de un archivo almacenado en disco.
- ✓ Función [readfile](#): para cargar el archivo en cuestión (html, imagen, documento PDF, etc.) y enviarlo directamente al navegador web.

Los diferentes archivos a servir deberán estar en una carpeta del proyecto llamada **data**. En la sección "*Información de interés*" tienes un archivo ZIP con los archivos que deben servirse en cada situación.

A la hora de realizar este ejercicio:

- ✓ Se evaluará el uso de arrays para simplificar el código (evitando el uso innecesario de estructuras `switch` o `if-elseif-else` echas a medida con todos los casos posibles).
- ✓ Se evaluará el uso de archivos php que contengan en un archivo separado la configuración.
- ✓ Se evaluará el uso de archivos php que contengan de forma separada las funciones necesarias.
- ✓ Se penalizará el hecho de uso de código innecesario.
- ✓ Se penalizarán los errores en el código tipo *warning*, *notice*, etc.

El siguiente vídeo ilustra el funcionamiento del ejercicio:



00:00

02:02

## Tarea online

### 1.- Descripción de la tarea

1.1.- Ejercicio 1:  
*jugando con los tipos mime*

1.2.- Ejercicio 2: *un texto brillante*

1.3.- Ejercicio 3: *solo personal autorizado*

1.4.- Ejercicio 4:  
*vamos paso a paso*

### 2.- Información de interés

### 3.- Evaluación de la tarea

### Anexo. Licencia de recursos

## 1.2.- Ejercicio 2: un texto brillante

En este ejercicio deberás elaborar un script llamado `ejercicio2.php` donde usarás las cookies para hacer un efecto de texto (importante: no se puede usar JavaScript para simular este efecto). Cada vez que se invoque el script `ejercicio2.php` se resaltará un carácter diferente de una cadena de texto mostrada. Por ejemplo, en una primera invocación se resaltará el primer carácter:

b r i l l a n t e

La segunda vez que se invoca este script, se resaltará el siguiente carácter:

b r i l l a n t e

Así sucesivamente hasta llegar al último carácter, momento en el cual se comenzará desde el principio. La página se recargará automáticamente cada 5 segundos, de forma que el carácter irá cambiando:

b r i l l a n t e

Para recargar la página automáticamente puedes usar la siguiente etiqueta meta: `<meta http-equiv="refresh" content="5">`

Además, en la página web habrá un pequeño formulario para poder cambiar la palabra a mostrar, a través de una caja de texto junto a un botón nombrado como "Siguiente". El funcionamiento será tal y como se describe en la continuación:

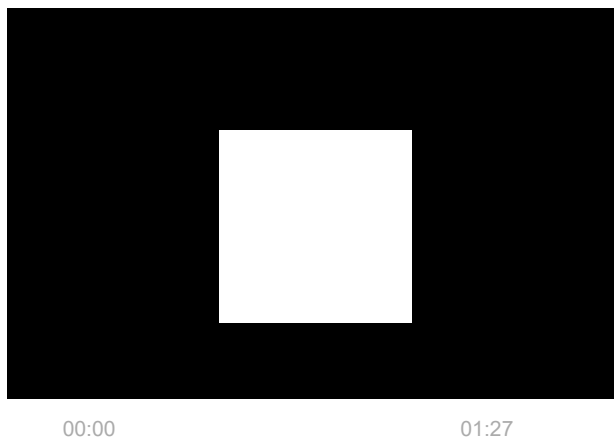
- ✓ Si se indica un texto en la entrada de texto y se hace clic en el botón "Siguiente", se cambiará el texto mostrado, comenzando a resaltar la primera letra del texto. Ten en cuenta que el texto a mostrar se cambiará solo si la longitud del mismo está entre 4 y 10 caracteres. En caso contrario debe mostrar una palabra por defecto a tu elección (`brillante` por ejemplo).
- ✓ Si no se indica ningún texto en la caja, al pulsar "Siguiente" simplemente se procederá a marcar el siguiente carácter de la palabra sin esperar que la página se recargue automáticamente.

Para realizar este ejercicio tendrás que usar cookies obligatoriamente, do para ser exactos:

- ✓ Cookie **varf**. En esta cookie se almacenara un **array serializado** con la siguiente información:
  - ➡ texto que se muestra.
  - ➡ posición del texto resaltada.
  - ➡ un número aleatorio mayor de 999999 (que se cambiará en cada consulta).
  - ➡ marca de tiempo (generada con el método `time()`).
- ✓ Cookie **verf**. Esta cookie contiene un resumen hash sha256 del contenido de la cookie **varf**. El objetivo es verificar que la cookie **varf** no ha sido manipulada en el lado del cliente, por lo que antes de proceder a usar los datos de la cookie **varf** debes realizar las comprobaciones oportunas para comprobar que dicha cookie no ha sido modificada.

El objetivo del número aleatorio y la marca de tiempo en la cookie **varf** es garantizar que cada vez que se consulta el script `ejercicio2.php` se genera un resumen hash sha256 diferente y no reutilizable en la cookie **verf**.

El siguiente vídeo ilustra el funcionamiento espeado de este ejercicio:



A la hora de realizar este ejercicio:

- ✓ Se penalizará el hecho de uso de código innecesario.
- ✓ Se penalizarán los errores en el código tipo *warning*, *notice*, etc.

## Debes conocer

El proceso de serialización de un dato en php consiste en obtener una representación portable de los datos almacenados en una variable (puede ser un array, un objeto, una cadena, etc.). La serialización de un array o de otro tipo de dato en php se realiza con el método [serialize](#).

```
$array = array('nombre' => 'Flash',  
               'profesion' => 'Superheroe',  
               'salario' => 'lo hace gratis');  
  
echo serialize ($array);
```

El código anterior generaría una cadena de texto como la siguiente:

```
a:3:{s:6:"nombre";s:5:"Flash";s:9:"profesion";s:10:"Super
```

El texto anterior se puede "deserializar" con el método [unserialize](#). Al deserializar obtenemos de nuevo el contenido del array u objeto perfectamente manipulable desde PHP:

```
$array=unserialize ('a:3:{s:6:"nombre";s:5:"Flash";s:9:"profesi  
echo $array['nombre'];  
echo $array['profesion'];  
echo $array['salario'];
```

## Debes conocer

Para generar un timestamp (marca de tiempo) en PHP deberás hacer uso de la función [time](#). Esta función genera un número entero con el número de segundos que han pasado desde 1 de enero de 1970 hasta el día de hoy.

## Tarea online

### 1.- Descripción de la tarea

1.1.- Ejercicio 1: jugando con los tipos mime

1.2.- Ejercicio 2: un texto brillante

1.3.- Ejercicio 3: solo personal autorizado

1.4.- Ejercicio 4: vamos paso a paso

### 2.- Información de interés

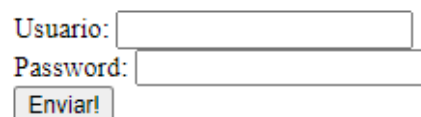
### 3.- Evaluación de la tarea

### Anexo. Licencia de recursos

## 1.3.- Ejercicio 3: solo personal autorizado

En este tercer ejercicio deberás implementar 3 scripts php (`ejercicio3.php`, `ejercicio3datos.php` y `ejercicio3salir.php`). La idea de este ejercicio es hacer uso de las sesiones para realizar un pequeño control de acceso que permita visualizar una información concreta solo a un conjunto de usuarios permitidos

Para simplificar la complejidad del ejercicio, los datos de autenticación van a estar almacenados en un archivo de configuración a través de un array. En el código debes prever que podría haber múltiples usuarios con acceso a los datos en dicho archivo de configuración. Entre dichos usuarios deberás poner 2 usuarios concretos: `admin1` y `admin2` (con contraseñas `admin1` y `admin2`).



*Ejemplo de formulario mostrado en el script `ejercicio3.php`*

La responsabilidad de cada script es la siguiente:

- ✓ **ejercicio3.php:**
  - Este script es responsable de verificar los datos de usuario y contraseña. Una vez verificados, establecerá en la sesión el nombre de usuario validado y un sello de tiempo que permita saber cuánto tiempo hace que se conectó.
  - Si la validación es correcta este script mostrará un único enlace al script `ejercicio3datos.php` con el texto "ver datos". Si la validación no es correcta deberá mostrarse un texto informativo al usuario de error que se ha producido.
  - Debes tener en cuenta que el formulario de validación solo aparecerá en caso de que el usuario no se haya validado anteriormente. Esto quiere decir que si el usuario consulta de nuevo este script (`ejercicio3.php`) habiéndose validado previamente, deberán mostrarse directamente el enlace "ver datos" al script `ejercicio3datos.php` y no deberá aparecer el formulario. Para lograr esto debes hacer uso de la información almacenada en la sesión.
  - Debes también tener en cuenta que la sesión caducará a los 120 segundos desde la conexión. Si el usuario vuelve a acceder al script `ejercicio3.php` y la sesión ha caducado, deberá destruirse la sesión y deberá volver a mostrarse el formulario solicitando el nombre de usuario y la contraseña.
- ✓ **ejercicio3datos.php:**
  - Este script es el responsable de cargar un archivo de datos CSV con datos personales de varias personas (descargable en la sección información de interés) y mostrarlos al usuario en forma de una tabla adecuada. Después de la tabla, habrá un enlace a `ejercicio3salir.php` con el texto "Salir".

## Bienvenido admin1

Nombre	Apellidos	Teléfono	E-mail	Cargo	Compañía
Pedro	Piedra	998288831	pedro@piedrassl.es	Director	Piedras de Pedro S.L.
Julia	Naranjo	828388481	julia@naranjoss1.es	Directora	Naranjos de Julia S.L.
Sebastian	Mesa	282883813	sebastian@mesass1.es	Director	Mesas de Sebastian S.L.
Maria del Mar	Puentes	828828182	mariadelmar@puentess1.es	Directora	Puentes de Maria del Mar S.

[Salir](#)

*Ejemplo de salida del script ejercicio3datos.php*

- Ten en cuenta que los datos no deberán mostrarse si el usuario no se ha validado correctamente o si su sesión ha caducado (caduca a los 120 segundos desde la autenticación). Para comprobar si el usuario se ha validado previamente debes utilizar los datos almacenados en la sesión, donde estará el nombre de usuario y el sello de tiempo.
  - Si la sesión ha caducado cuando se consulta este script, deberás destruir la sesión y no mostrar los datos al usuario (ni el enlace "Salir").
  - Si la sesión ha caducado o el usuario no es correcto (porque el usuario a validar ha desaparecido del archivo de configuración de usuarios), puedes elegir entre hacer una de las siguientes dos cosas:
    - Mostrar un enlace al script `ejercicio3.php` al usuario con el texto "Volver".
    - Redireccionar la petición al script `ejercicio3.php` con una cabecera HTTP tipo `Location`. Esta opción requiere el uso de la función [header](#).
- ✓ **ejercicio3salir.php:**
- Este script solo hará una cosa: destruir la sesión. Después de destruir la sesión puedes elegir entre una de estas dos acciones:
    - Mostrar un enlace al script `ejercicio3.php` al usuario con el texto "Volver".
    - Redireccionar la petición al script `ejercicio3.php` con una cabecera HTTP tipo `Location`. Esta opción requiere el uso de la función [header](#).

**Nota: normalmente las sesiones se "regeneran" aumentando el tiempo de validez de la sesión una vez que el usuario realiza alguna acción. No es algo que haya que hacer en este ejercicio.**

Por último, ten en cuenta que a la hora de realizar este ejercicio:

- ✓ Se evaluará el uso de arrays para simplificar el código (evitando el uso innecesario de estructuras `switch` o `if-elseif-else` echas a medida para todos los casos posibles).
- ✓ Se evaluará el uso de archivos php que contengan en un archivo separado la configuración.
- ✓ Se evaluará el uso de archivos php que contengan de forma separada las funciones necesarias.
- ✓ Se evaluará que se separen los formularios HTML (y otros fragmentos HTML) en archivos separados.
- ✓ Se penalizará el hecho de uso de código innecesario.
- ✓ Se penalizarán los errores en el código tipo *warning*, *notice*, etc.



## Debes conocer

Para manejar archivos tipo CSV (Comma Separated Values), PHP tiene dos funciones: [fputcsv](#) y [fgetcsv](#). La primera se utiliza para escribir líneas en el archivo CSV y la segunda para obtener líneas del archivo CSV. Su funcionamiento es bastante sencillo:

- ✓ Abrimos el archivo con [fopen](#) en modo lectura ('r') o escritura ('w') dependiendo de la operación que vayamos a hacer.
- ✓ Leemos o escribimos líneas CSV en el archivo usando las funciones correspondientes. Para que los datos no den problemas, es conveniente que uses la función [addslashes](#) antes de escribir los datos y [stripslashes](#) después de leerlos.
- ✓ Cerramos el archivo con [fclose](#) una vez que hemos realizado las operaciones.

Para conocer más sobre el formato CSV consulta el siguiente enlace:

[Valores separados por comas \(CSV\)](#)

## Tarea online

### 1.- Descripción de la tarea

1.1.- Ejercicio 1:  
jugando con los  
tipos mime

1.2.- Ejercicio 2: un  
texto brillante

1.3.- Ejercicio 3: solo  
personal autorizado

1.4.- Ejercicio 4:  
vamos paso a paso

### 2.- Información de interés

### 3.- Evaluación de la tarea

### Anexo. Licencia de recursos

## 1.4.- Ejercicio 4: vamos paso a paso

En este ejercicio vas a realizar un formulario que vaya pidiendo los siguientes datos de usuario progresivamente:

- ✓ En el paso 1 se pedirán:
  - ➡ Nombre.
  - ➡ Apellidos.
- ✓ En el paso 2 se pedirán:
  - ➡ Teléfono.
  - ➡ Email.
- ✓ En el paso 3 se pedirán:
  - ➡ Cargo en la empresa.
  - ➡ Nombre de la empresa.
- ✓ En el paso 4 no se pedirá ningún dato, solo se almacenarán los datos en el archivo CSV (el mismo que el usado en el ejercicio 3).

A continuación tienes ejemplos de las diferentes interfaces:

### Paso 1. Nombres y apellidos.

Nombre:   
Apellidos:

[Ir a paso 2](#)

*Ejemplo de formulario del paso 1.*

### Paso 2. Teléfono e email.

Nombre: Salvador  
Apellidos: Romero Villegas

Teléfono:   
Email:

[Ir a paso 3](#)

[Volver a paso 1](#)

*Ejemplo de formulario del paso 2.*

### Paso 3. Puesto y empresa.

Nombre: Salvador  
Apellidos: Romero Villegas  
Teléfono: 999999999  
Email: unomas@salvadores.es

Puesto:   
Empresa:

*Ejemplo de formulario del paso 3.*

### Paso 4. Confirmación de registro.

Nombre: Salvador  
Apellidos: Romero Villegas  
Teléfono: 999999999  
Email: unomas@salvadores.es  
Puesto: profesor  
Empresa: IES Aguadulce

## ¡¡Se registró tu solicitud para el evento!!

### Un agente se pondrá en contacto contigo

*Ejemplo de interfaz del paso 4.*

Para realizar este ejercicio deberás tener en cuenta lo siguiente:

- ✓ Se debe implementar un único archivo php llamado `ejercicio4.php` aunque debes usar tantos archivos adicionales como necesites (que incorporarás usando `include`, `include_once`, `readfile`, etc.).
- ✓ Deberás hacer uso de sesiones almacenar temporalmente los datos. Esto quiere decir que no debe enviarse información de un formulario a siguiente o al anterior en campos `hidden` o similar. Puedes usar campos `hidden` para otros propósitos, pero **no para enviar datos que debería estar almacenados en la sesión**.
- ✓ Para simplificar el ejercicio, todos los datos se validarán y sanearán de la misma forma, realizando un procesamiento básico:
  - ➡ Validación: longitud mínima de 2 caracteres y máxima de 30.

- ➡ Saneado: se usa la función `addslashes` sobre cada campo.
- ✓ Importante: si los datos de un paso no son válidos, no debe poder avanzarse al siguiente paso. Se mostrará el problema al usuario y se permanecerá en el mismo paso.
- ✓ Cuando un usuario accede a `ejercicio4.php` se debe mostrar la interfaz correspondiente al paso por el que se quedó la última vez que accedió (información que debe estar recogida en la sesión). Es decir, si se abandona la página en el paso 2 y luego se vuelve a acceder después de un tiempo usando el mismo navegador, debe volver a mostrar el paso 2 y no otro paso.
- ✓ En cada paso (excepto el 4), debe existir la opción de volver a un paso anterior (del paso 3 al paso 2 y del paso 2 al paso 1). Cuando los datos se han rellenado previamente en un paso posterior (por ejemplo: se rellenaron en el paso 2 y se volvió al paso 1), los datos de pasos posteriores no deben borrarse (simplemente permite editarlo nuevamente antes finalizar el proceso).
- ✓ Cuando el usuario alcanza el penúltimo paso (paso 3), aparecerá un botón con el texto "Solicitar!" que llevará al paso 4. Si se hace clic en el botón "Solicitar!" se llevará al paso 4 y se guardarán los datos recogidos en los pasos anteriores en un archivo CSV (el mismo archivo usado en el ejercicio 3).
- ✓ En el paso 4 aparecerá también un botón con el texto "¿Registrar otra solicitud?". Si se hace clic en dicho botón se comenzará el proceso desde el paso 1.

Por último, ten en cuenta que a la hora de realizar este ejercicio:

- ✓ Se evaluará el uso de arrays para simplificar el código (evitando el uso innecesario de estructuras `switch` o `if-elseif-else` echas a medida con todos los casos posibles).
- ✓ Se evaluará el uso de archivos php que contengan en un archivo separado la configuración.
- ✓ Se evaluará el uso de archivos php que contengan de forma separada las funciones necesarias.
- ✓ Se evaluará que se separen los formularios HTML (y otros fragmentos HTML) en archivos separados.
- ✓ Se penalizará el hecho de uso de código innecesario.
- ✓ Se penalizarán los errores en el código tipo *warning*, *notice*, etc.

## Tarea online

### 1.- Descripción de la tarea

1.1.- Ejercicio 1: jugando con los tipos mime

1.2.- Ejercicio 2: un texto brillante

1.3.- Ejercicio 3: solo personal autorizado

1.4.- Ejercicio 4: vamos paso a paso

### 2.- Información de interés

### 3.- Evaluación de la tarea

### Anexo. Licencia de recursos

## 2.- Información de interés

### Recursos necesarios y recomendaciones

### Recomendaciones específicas de esta tarea

- ✓ El uso de sesiones y cookies puede ser sumamente lioso, por lo que es altamente recomendable que uses y configures el depurador de NetBeans o Visual Studio Code.
- ✓ En muchas instalaciones de php los errores están configurados para no mostrarse al usuario. Mientras estás desarrollando una aplicación es imprescindible que los errores de php se muestren, de forma que podamos corregir cualquier aplicación antes de que llegue al usuario final. En el siguiente enlace se explica como:

[¿Cómo mostrar errores php?](#)

- ✓ En esta tarea no se especifica una estructura de proyecto, se ha dejado libertad a ese respecto. Intenta ordenar los archivos de la forma más adecuada para que resulte un código claro y fácil de mantener.

### Recursos para esta tarea

En esta tarea deberás hacer uso de los siguientes archivos:

[Recursos necesarios para realizar la tarea](#) (zip - 118,91 KB).

### Recomendaciones generales

- ✓ Como ya sabes, para escribir aplicaciones en PHP necesitarás un entorno XAMPP, LAMPP o similar. No te vamos a pedir que instales un entorno u otro, o que trabajes con un sistema operativo concreto, pero sí que todos los **archivos vayan codificados en UTF-8** y que **las rutas a los archivos sean siempre relativas**.
- ✓ Como plataforma de desarrollo (para escribir código), te aconsejamos **NetBeans 12**. Es la misma herramienta que se usa en otros módulos, no tendrás que cambiar de herramienta.
- ✓ No abordes la tarea hasta que hayas repasado todos los contenidos, hecho al menos uno de los intentos para el examen en línea, de forma que hayas tenido ocasión de consultar y resolver en los foros cualquier duda que te haya podido surgir.

## Indicaciones de entrega

Una vez realizada la tarea, el envío se realizará a través de la plataforma. Comprime la carpeta del proyecto en un fichero .zip y nómbralo siguiendo las siguientes pautas:

**Apellido1\_Apellido2\_Nombre\_DWES\_Tarea03**

## Tarea online

### 1.- Descripción de la tarea

**1.1.- Ejercicio 1:**  
*jugando con los tipos mime*

**1.2.- Ejercicio 2:**  
*un texto brillante*

**1.3.- Ejercicio 3:**  
*solo personal autorizado*

**1.4.- Ejercicio 4:**  
*vamos paso a paso*

### 2.- Información de interés

### 3.- Evaluación de la tarea

### Anexo. Licencia de recursos

## 3.- Evaluación de la tarea

### Criterios de evaluación implicados

Resultado de aprendizaje 4: desarrolla aplicaciones Web embebidas en lenguajes de marcas analizando e incorporando funcionalidades según especificaciones.

#### ✓ Criterios de evaluación según normativa:

- Se han identificado los mecanismos disponibles para el mantenimiento de la información que concierne a un cliente Web concreto y se han señalado sus ventajas.
- Se han utilizado sesiones para mantener el estado de las aplicaciones Web.
- Se han utilizado «cookies» para almacenar información en el cliente Web y para recuperar su contenido.
- Se han identificado y caracterizado los mecanismos disponibles para la autenticación de usuarios.
- Se han escrito aplicaciones que integren mecanismos de autenticación de usuarios.
- Se han utilizado herramientas y entornos para facilitar la programación, prueba y depuración del código.



[Peggy\\_Marco](#) (Pixabay License)

### ¿Cómo valoramos y puntuamos tu tarea?

En esta puedes observar la puntuación máxima asignada a cada ejercicio de la tarea:

#### Rúbrica de la tarea

Se ha implementado el ejercicio 1 apropiadamente y

hasta

1,75

funciona correctamente.	<b>puntos</b>	
Se ha implementado el ejercicio 2 apropiadamente y funciona correctamente.	<b>hasta puntos</b>	<b>2,25</b>
Se ha implementado el ejercicio 3 apropiadamente y funciona correctamente.	<b>hasta puntos</b>	<b>2,75</b>
Se ha implementado el ejercicio 4 apropiadamente y funciona correctamente.	<b>hasta puntos</b>	<b>3,25</b>