UT 6: SESIONES Y COOKIES

1. **Introducción**
2. Cookies
3. Sesiones
4. Diferencias

1. INTRODUCCIÓN

La entrada de información en un sistema es uno de sus puntos **vulnerables**.

Para controlar el acceso a una aplicación web usando PHP es necesario mandar los datos a un archivo que procesa y validar al usuario que intenta logarse en el sistema accediendo a una base de datos MySQL.

Generalmente, por motivos de usabilidad, un sitio web tendrá varios formularios. Entre los distintos formularios se utilizarán algunos **datos comunes**.

Normalmente, es necesario conservar los valores que se han introducido en el proceso de autenticación o algo que determine que se autenticado correctamente.

A día de hoy lo que se almacena es un token (secuencia de caracteres y números) que determinan que el usuario se ha autenticado correctamente.

1. INTRODUCCIÓN

El control de usuarios de una aplicación en Internet es uno de los procedimientos más frecuentes y comunes, de forma que sólo puedan acceder a determinadas páginas aquellas personas que sean reconocidas y estén autorizadas en sus diferentes perfiles. Para ello, se van a utilizar dos técnicas:

- **Cookies**: que serán almacenadas en el cliente.
- **Sesiones**: que serán almacenadas en el servidor.

UT 6: SESIONES Y COOKIES

1. Introducción
2. **Cookies**
3. Sesiones
4. Diferencias

2. COOKIES

Las cookies se pueden definir como un fragmento de información **almacenado** por un navegador web durante la visita de un usuario a una página web.

La información guardada tiene forma de fichero de texto y suele contener datos que en algún momento han sido preguntados al usuario de la página, por ejemplo, su dirección de correo electrónico o su teléfono.

Los propósitos fundamentales de usar cookies son:

- conocer los hábitos de navegación del usuario
- recordar accesos a la web.

En la actualidad, cuando se visita un determinado sitio, atendiendo a la GDPR (normativa que regula la protección de los datos de los ciudadanos que viven en la Unión Europea), es necesario que el usuario acepte cuáles son las cookies que desea.

2. COOKIES

Podemos diferenciar los siguientes tipos de cookies:

1. Temporales: conservan su valor hasta que el navegador se cierra.
2. Persistentes: son conservadas en el cliente con el objeto de poder ser utilizadas en diferentes sesiones.
3. Propias: diseñadas por una determinada página web y propiedad de la misma
4. De terceros: pertenecen a empresas externas a las páginas que piden su inserción cuando navegas por una determinada web.
5. Técnicas: su funcionalidad es necesaria para optimizar el funcionamiento de la web.

2. COOKIES

6. Preferencias: permiten almacenar datos como el tipo de navegador que usas o el idioma. Es decir, conservan tanto configuraciones como preferencias.

7. Rendimiento y análisis: permiten que se puedan recoger datos analíticos sobre la experiencia en un sitio web, por ejemplo, si un determinado proceso de compra no se llevó a cabo.

8. Publicitarias: cuya funcionalidad es hacer un chequeo del comportamiento del usuario durante el proceso de navegación; de esta forma es posible crear un perfil de los gustos e intereses y mostrar publicidad relevante.

2. COOKIES

Los diferentes navegadores del mercado tienen funcionalidades que permiten administrar las cookies almacenadas en el sistema.

En el caso de Chrome, por ejemplo, para poder acceder a nuestras preferencias sobre las cookies los pasos que hemos de seguir son:

1. Abrir Chrome.

2. En la esquina superior derecha, haz clic en Más | > Configuración.

3. En Privacidad y seguridad, haz clic en Cookies y otros datos de sitios.

4. A continuación, realiza la acción que desees sobre las cookies: borrar, permitir..

2. COOKIES

- Cuando se crea una cookie en el servidor es enviada al cliente en la cabecera html.
- En las siguientes peticiones que realice ese cliente se enviará al servidor la cookie creada.

2. COOKIES

La **información de una cookie** está definida en el protocolo HTTP e integrada por los siguientes datos:

1. **Nombre** de la cookie.
2. **Valor**, que es el contenido de la cookie.
3. **Fecha de caducidad**: periodo de vigencia durante el cual puede ser recuperada por el servidor.
4. **Dominio** dentro del cual es válida la cookie. Si no se especifica, sólo es válida para el dominio del servidor que la generó. El navegador devuelve la cookie a cualquier equipo que tenga el dominio especificado.
5. **Ruta**: por defecto es "/", es decir, el directorio raíz de las páginas a las que se devuelve la cookie.
6. **Seguro**: por defecto está desactivado (Disabled), pero puede especificarse este parámetro para establecer un canal seguro del protocolo HTTPS si se necesita por tratarse de una información confidencial.

2. COOKIES

CREAR COOKIE

La función setcookie() crea y envía al cliente una cookie estableciendo los elementos que la integran. Su sintaxis es la siguiente:

```
setcookie ("nombre de la cookie", $valor, fecha de caducidad,
ruta, dominio, seguro);
```

De estos seis parámetros sólo es obligatorio el primero, nombre de la cookie.

2. COOKIES

La función `setcookie()` crea y envía una cookie al cliente juntamente con la cabecera de la página. Por eso, es **absolutamente imprescindible** escribir esta función antes que las etiquetas <HTML> o <HEAD>

Hay que tener en cuenta que, cuando se crea una cookie, es enviada al cliente, por lo cual no se puede ver hasta que se carga de nuevo la página.

2. COOKIES

```
$usuario="Jorge";
setcookie("Nombre", $usuario);


setcookie("Nombre","Jorge");
```

2. COOKIES

Para establecer caducidad:

En el tercer parámetro indicamos que dure 5 minutos (300 segundos) a partir del momento en que se crea

```
$pieza="Tornillo";
setcookie("Hierro", $pieza, time()+300);
```

Para que dure hasta las 12:30:00 del día 31 de julio de 2031:

```
setcookie("Hierro", $pieza, mktime(12,30,0,7,31,2021);
```

2. COOKIES

Para acceder a una cookie se usa el array $_COOKIE.

Por otro lado, se pueden establecer arrays de cookies, usando la notación de array. Esto es útil para organizar cookies relacionadas entre sí.

```
setcookie(“listaCompra[0]”,”agua”)
setcookie(“listaCompra[1]”,”pan”)
```

2. COOKIES

ELIMINAR COOKIE

Para eliminar una cookie basta escribir la función setcookie() incluyendo como único argumento el nombre de la cookie que se quiere eliminar.

Por ejemplo, la instrucción setcookie("Nombre"); borra la cookie del mismo nombre.

2. COOKIES

MODIFICAR COOKIE

Para modificar una cookie basta escribir de nuevo la función setcookie() incluyendo los nuevos argumentos y usando el nombre de la cookie que se quiere editar.

UT 6: SESIONES Y COOKIES

1. Introducción
2. Cookies
3. **Sesiones**
4. Diferencias

3. SESIONES

$_SESSION es un array especial utilizado para **retener información** a lo largo de las solicitudes que un usuario realiza durante su interacción con un sitio web o aplicación. La información almacenada en una sesión puede ser recuperada en cualquier momento mientras la sesión esté activa.

A cada usuario que visita un sitio web, se le asigna un identificador único denominado **session ID**. Este identificador permite identificar la sesión y asegura que la información esté disponible exclusivamente para ese usuario en particular.

Para garantizar la seguridad de las sesiones, se recomienda almacenar solo este session ID en el cliente, mientras que cualquier información asociada a la sesión se guarda en el servidor de manera segura. Este enfoque minimiza el riesgo de exposición de datos sensibles y mejora la protección de la privacidad del usuario.

3. SESIONES

Parámetros de configuración de las sesiones:

- En el archivo php.ini se pueden configurar los parámetros de sesion.
- También se pueden configurar de forma dinámica invocando a la función ini_set(parametro, valor)
- Algunos de estos parámetros:
  - session.gc_maxlifetime: Nº de segundos a partir de los cuales la sesión se considera basura.
  - session.gc_probability: Junto con session.gc_divisor para controlar la probabilidad de que arranque la rutina gc (garbage colector).

3. SESIONES

ABRIR UNA SESIÓN Y CONOCER LOS DATOS DE LA MISMA

La función session_start() abre una nueva sesión de trabajo o prolonga la que ya está abierta conservando su identificador. Su sintaxis es sencilla:

```
session_start();
```

3. SESIONES

Diversas funciones nos permiten acceder a los datos de la sesión abierta.

- La función **session_name()** devuelve y/o establece el nombre de la sesión abierta en ese momento.
- La función **session_save_path()** devuelve y/o establece el camino donde se guardan el fichero con los datos de un identificador de sesión.
- La función **session_id()** devuelve y/o establece el nombre del identificador de la sesión en curso.

3. SESIONES

REGISTRAR INFORMACIÓN DE UNA SESION

Se usa el array asociativo $_SESSION.

```
$_SESSION['variable'] = valor;
```

ACCEDER AL CONTENIDO DE UNA SESION

También se usa el array asociativo $_SESSION.

Consultar si una variable de sesión está establecida

```
isset($_SESSION['variable'])
```

Mostrar el contenido de una variable establecida

```
echo $_SESSION['variable']
```

EJEMPLO DE USO DE SESIONES

```
<?php

session_start();

$_SESSION["usuario"]="Ana";

$_SESSION["hora"]=time();

$_SESSION["pagina"]="$_SERVER["PHP_SELF"];

if (empty($_SESSION["contador"])) {

     $_SESSION["contador"]=0;

}

$_SESSION["contador"]++;

echo "Nombre de la sesión: ".$_SESSION['nombre'].
"<P>Nombre del usuario : ".$_SESSION['usuario'].
"<P>Hora última entrada: ".strftime("%H:%M:%S

del %d/%m/%Y",$_SESSION['hora']);

?>
```

3. SESIONES

ELIMINAR VARIABLES ESTABLECIDA EN UNA SESION

Para eliminar una variable de sesión

```
unset($_SESSION['variable']);
```

Para eliminar todas las variables que existan en esa sesión

```
session_unset();
```

Después de cualquiera de estos dos comandos, se pueden seguir guardando variables de sesión en esta sesión. Para dejar de utilizar esta sesión, hay que destruirla.

3. SESIONES

DESTRUIR UNA SESIÓN

La función session_destroy() borra toda la información que se haya recogido y asociado a la sesión activa.

```
session_destroy();
```

3. SESIONES

SEGURIDAD EN LAS SESIONES

Sessión Time-Outs : Establecer un tiempo de vida a las sesiones es algo importante para lidiar con las sesiones de usuarios en aplicaciones. Si un usuario inicia sesión en un lugar y se olvida de cerrarla, otros pueden continuar con la misma, por eso es mejor establecer un tiempo máximo de sesión.

3. SESIONES

El código asegura que si no hay ninguna actividad en 10 minutos, cualquier request en adelante redigirirá a la página de logout.

```
session_start();
$inactividad = 600;
if(isset($_SESSION["timeout"])){
      //Calcular el tiempo de vida de la sesión (TTL = Time To Live)
      $sessionTTL = time() - $_SESSION["timeout"];
      if($sessionTTL > $inactividad){
          session_destroy();
          header("Location: /logout.php");
      }
 }
 // El siguiente key se crea cuando se inicia sesión
 $_SESSION["timeout"] = time();
```

3. SESIONES

SEGURIDAD EN LAS SESIONES

Regenerar el Session ID: La función `session_regenerate_id()` crea un nuevo ID único para representar la sesión actual del usuario. Esto se debe realizar cuando se realizan acciones importantes como logearse o modificar los datos del usuario. Darle a las sesiones un nuevo ID reduce la probabilidad de ataques session hijacking.

```
session_start();
if($_POST["usuario"] = "admin" && $_POST["password"] == sha1($password)){
    $_SESSION["autorizado"] = true;
    session_regenerate_id();
}
```

3. SESIONES

SEGURIDAD EN LAS SESIONES

Destruir la sesión: Es importante utilizar session_*destroy()* cuando ya no se vaya a hacer uso de la sesión.

Utilizar almacenamiento permanente: Utilizar una base de datos para almacenar los datos que se cree que serán persistentes. Dejarlos en la sesión por demasiado tiempo abre de nuevo puertas a ataques. Depende del desarrollador decidir que datos serán almacenados o se mantendrán en $_SESSION.

UT 6: SESIONES Y COOKIES

1. Introducción
2. Cookies
3. Sesiones
4. **Diferencias**

4. DIFERENCIAS

Existen algunas diferencias entre sesiones y cookies:

- Ubicación. Aunque el usuario deshabilite las cookies en el navegador, esto no supone ningún problema para las sesiones. La información de una sesión se conserva en el servidor.
- Persistencia. La información de las sesiones, a diferencia de las cookies, solo es accesible durante el proceso de navegación.