TEMA

3

2ºDAW

PHP y las Base de datos

1 Introducción a las BD

Una **Base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso a través del gestor.

Un **sistema gestor de base de datos** SGBD (DataBase Managenent System-DBMS) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarios para el almacenamiento y búsqueda de la información del modo más eficiente posible. En la actualidad, existen multitud de SGBD en la mayoría relacionales (Tablas – registros- campos y relaciones). Los más utilizados son

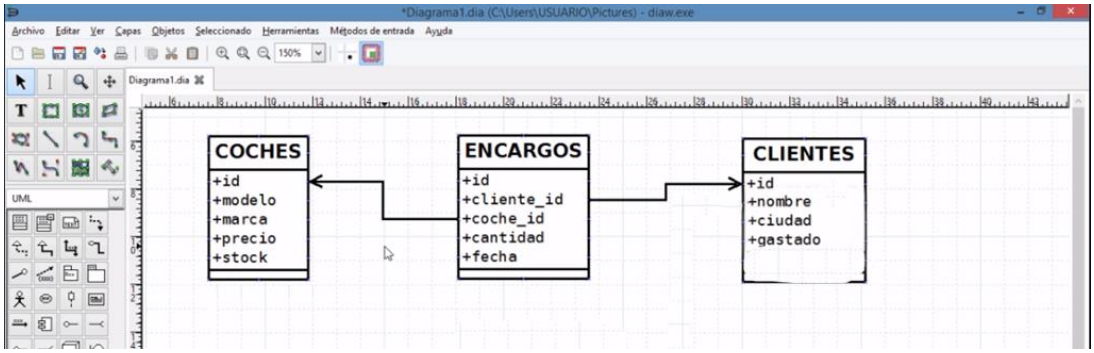
- MySQL / MariaDB
- Microsoft SQL Server
- Oracle
- Microsoft Access
- PostgreSQL

Ejercicios

✓ **Ejercicio 1:** Diseñar y crear una base de datos de un concesionario de coches con la relación M:M entre coches y clientes.

- **Crea el diseño de tablas utilizando el programa DIA u otro programa similar**

Coche	Encargos	Clientes
id – int(10) **Primary key not null**	id – **int**(10) Primary key not null	id – INT(10) Primary key not null
Modelo – varchar(100)	cliente_id - **int**(10) Clave foránea	Nombre – varchar (100)
Marca- varchar(50)	Coche_id - **int**(10) Clave foranea	Ciudad – varchar(100)
Precio – int(20)	Cantidad - **int**(20)	Gastado – float (50,2)
Stock – int(20)	Fecha - date	



- Comprueba que el usuario root tiene contraseña
- Crea la base de datos c**oncesionario** en phpmyadmin con cotejamiento utf8_unicode_ci
- Crea un usuario (sin permiso de administración) utilizando **phpmyadmin** para la base de datos concesionario.
- **Crear las tablas e inserta un par de registros con un script de SQL utilizando la siguiente foto como ejemplo (Utiliza constraint y ENGINE=InnoDB para la integridad referencial)**

```
CREATE TABLE IF NOT EXISTS encargos(
.................
..........
...............
CONSTRAINT pk_encargos PRIMARY KEY(id),
CONSTRAINT fk_encargo_cliente FOREIGN KEY(cliente_id) REFERENCES clientes(id),
CONSTRAINT fk_encargo_coche FOREIGN KEY(coche_id) REFERENCES coches(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish2_ci;
```

Nota: Utiliza la función **curdate()** dentro del INSERT para introducir la fecha actual en formato **AAAA-MM-DD**

- Introduce la ruta del mysql Windows en las variables de entorno Windows (PATH)
- Utiliza **mysqldum**p para importa el script por consola (mibase es el nombre de la BD creada en phpmyadmin).

```
mysql --password=miclave --user=miuser mibase < archivo.sql
```

- Entra en MYsql por consola: **mysql -u usuario -p** y comprueba que la base datos y las tablas se han creado correctamente
  - **Show databases;**
  - **Use nombreBD;**
  - **Show tables;**
  - **Describe nombre_tabla;**
- Utiliza PHPMyAdmin para insertar algunos registros más.

2 Acceso a BD con PHP

Desde una página con código escrito en PHP nos podemos conectar a una base de datos y ejecutar **comandos en lenguaje SQL** - para hacer consultas, insertar o modificar registros, crear tablas, dar permisos, etc.

PHP puede trabajar con la práctica totalidad de gestores de bases de datos que hay disponibles, tanto comerciales como de código abierto. Las formas de acceder a la BD son las siguientes:

- **Mysql:** La interfaz mysql permite acceder a bases de dados MySQL. Esta interfaz se considera obsoleta y se desaconseja su uso en aplicaciones nuevas.
- **Mysqli:** Se trata de una mejora de la interfaz mysql (la "i" viene de "improved"). Por ejemplo, la extensión mysqli añade ciertas funcionalidades que no tiene mysql como las transacciones, los procedimientos almacenados y las sentencias múltiples. En aplicaciones nuevas se recomienda usar esta API, o bien PDO. Para implementar este metodo tenemos 2 opciones: **la tradicional (programación estructurada) y la (orientada a objetos)**
- **PDO:** PDO son las siglas de PHP Data Objects (Objetos de Datos de PHP). Proporciona una capa de abstracción de tal forma que los métodos utilizados para acceder a los datos son independientes del sistema gestor de bases de datos utilizado. En la práctica, permite cambiar de SGBD sin cambiar el código PHP.

Extensión Mysqli

Conexión al MySQL

Creamos una nueva conexión con el servidor MySQL en el host que se indicando el servidor, usuario y contraseña.

Nota: Primera opción es por procedimientos y la segunda orientada a objetos

- **$conexion = mysqli_connect($servidor, $usuario, $clave, $bd)**;
- **$conexion = new mysqli($servidor,$usuario,$clave, $bd);**

La conexión se cerrará automaticamente cuando finalice el script. Para cerrar la conexión antes

Mysql_close($conexion); // $conexion ->close();

PHP dispone de dos funciones para ver si la sentencia se ha ejecutado correctamente

- **mysqli_connect_errno() / $conexion->connect_errno :** Devuelve 0 si no hay error y si no un número con el error correspondiente
- **mysqli_connect_error()/ $conexion->connect_error:**Devuelve la descripción del error

Programación estructurada (conecta.php)

```php
<?php
$servidor="localhost";
$usuario="user";
$clave= "pestillo";
$bd="concesionario";
//Realizamos la conexión al servidor y guarda los parametros en la variable conexion
$conexion=mysqli_connect($servidor,$usuario,$clave,$bd);
if (mysqli_connect_errno()) {
echo "No se ha podido establecer conexión con el servidor de bases de datos.<br/>";
die ("Error: " . mysqli_connect_error());
}
//consulta para configurar la codificación de caractéres
mysqli_query($conexion, "SET NAMES 'utf8'");
?>
```

Orientado a objetos (conectaOO.php)

```php
<?php
$servidor="localhost";
$usuario="user";
$clave= "pestillo";
$bd="concesionario";
//Realizamos la conexión al servidor y guarda los parametros en la variable conexion
$conexion=new mysqli($servidor,$usuario,$clave,$bd);
if ($conexion->connect_errno) {
echo "No se ha podido establecer conexión con el servidor de bases de datos.<br/>";
die ("Error: " . $conexion->connect_error);
}
//consulta para configurar la codificación de caractéres
$conexion->set_charset("utf8");
?>
```

PHP Data Object – PDO (conectaPDO.php)

```php
<?php
$servidor="localhost";
$usuario="user";
$clave= "pestillo";
$bd="concesionario";

try {
// mysql es el gestor de Base de datos
   $conn = new PDO("mysql:host=$servidor;dbname=$bd", $usuario, $clave);
   // Establece los atributos de los reportes de errores
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   echo "Conexión satisfactoria";
   }
catch(PDOException $e)
   {
   echo ( "Error de conexión: " . $e->getMessage());
   }
?>
```

Una de las ventajas de utilizar PDO es que tiene una clase de Excepción que pueda ocurrir en cualquier consuta. Si se produce una excepción el script deja de ejecutarse

Listado completo de una tabla

- **$fila= mysqli_fetch_assoc($resultado) // $fila= $resultado->fetch_assoc():** Devuelve un array asociativo que corresponde a la fila recuperada o FALSE si no hay más filas.
- **mysqli_num_rows($resultado) //$resultado->num_rows:** Devuelve el número de filas

Programación estructurada (listar.php)

```
<!DOCTYPE html>

<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
    require_once 'conecta.php';
    $sql="select * from coches";
    $resultado= mysqli_query($conexion, $sql);
    if (mysqli_num_rows($resultado)>0){  ?>
    <table style="border:solid 1px">
      <caption>Coches del concesionario</caption>
      <tbody>
      <tr><th> Identificador</th>
      <th> Marca</th>
      <th> Modelo</th>
      <th> Precio</th>
      <th> Stock</th></tr>

      <?php while ($fila= mysqli_fetch_assoc($resultado) ){?>
      <tr><td><?= $fila['id']?></td>
      <td><?= $fila['marca']?></td>
      <td><?= $fila['modelo']?></td>
      <td><?= $fila['precio']?></td>
      <td><?= $fila['stock']?></td></tr>
      <?php
      }
      }else{
      echo '0 Registros';}
      mysqli_close($conexion);
    ?>
   </tbody>
    </table>
  </body>
</html>
```

Orientado a objetos (listarOO.php)

```
<!DOCTYPE html>

<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
    require_once 'conectaOO.php';
    $sql="select * from coches";
    $resultado= $conexion->query($sql);
    if ($resultado->num_rows>0){  ?>
    <table style="border:solid 1px">
      <caption>Coches del concesionario</caption>
      <tbody>
      <tr><th> Identificador</th>
      <th> Marca</th>
      <th> Modelo</th>
      <th> Precio</th>
      <th> Stock</th></tr>

      <?php while ($fila= $resultado->fetch_assoc() ){?>
      <tr><td><?= $fila['id']?></td>
      <td><?= $fila['marca']?></td>
      <td><?= $fila['modelo']?></td>
      <td><?= $fila['precio']?></td>
      <td><?= $fila['stock']?></td></tr>
      <?php
      }
      }else{
      echo '0 Registros';}
      $conexion->close();
      ?>
   </tbody>
    </table>
  </body>
</html>
```

PHP Data Object – PDO (listarPDO.php)

```
<!DOCTYPE  html>
<html>
<head>
<meta  charset="UTF-8">
</head>
<body>
<h2>
```

```
Base de datos <u>banco</u><br> Tabla <u>cliente</u><br>
</h2>
<?php require_once 'conectapdo.php';
  $consulta = $conn->query("select * from coches");
?>
<table style='border: solid 1px black;'>
<tr>
   <th>ID</th>
   <th>Marca</th>
   <th>Modelo</th>
   <th>Precio</th>
   <th>stock</th>
</tr>
<?php
while ($coche =  $consulta->fetchObject())  {
?>
<tr>
<td><?=  $coche->id ?></td>
<td><?=  $coche->marca ?></td>
<td><?=  $coche->modelo ?></td>
<td><?=  $coche->precio ?></td>
<td><?=  $coche->stock ?></td>
</tr>
<?php
}
?>
</table>
<br/>
Número de clientes: <?= $consulta->rowCount();  ?>
<?php  $conn=null;  ?>
</body>
</html>
```

Insertar registro

Programación estructurada (insertar.php)

```
<!DOCTYPE html>

<html lang="es">
   <head>
      <meta charset="UTF-8">
      <title></title>
   </head>
   <body>
      <?php
      require_once 'conecta.php';
     $modelo= $_POST['modelo'];
     $marca= $_POST['marca'];
    $precio=(int) $_POST['precio'];
   $stock=(int) $_POST['stock'];
```

```
    $sql="insert into coches values(null,'$modelo','$marca',$precio,$stock)";
     $insert= mysqli_query($conexion, $sql);
     if ($insert){
        echo "Datos insertados correctamente";
     }else {
        echo "Error:". mysqli_error($conexion);
     }
     mysqli_close($conexion);
     ?>
  </body>
</html>
```

Orientado a objetos (insertarOO.php)

```
<!DOCTYPE html>

<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
    require_once 'conectaOO.php';
   $modelo= $_POST['modelo'];
   $marca= $_POST['marca'];
  $precio=(int) $_POST['precio'];
  $stock=(int) $_POST['stock'];

    $sql="insert into coches values(null,'$modelo','$marca',$precio,$stock)";
     $insert= $conexion->query($sql);
     if ($insert){
        echo "Datos insertados correctamente";
     }else {
        echo "Error:". $conexion->connect_error;
     }
     $conexion->close();
     ?>
  </body>
</html>
```

PHP Data Object – PDO(insertarPDO.php)

```
<?php
require_once 'conectapdo.php';
try {
    $modelo= $_POST['modelo'];
    $marca= $_POST['marca'];
   $precio=(int) $_POST['precio'];
  $stock=(int) $_POST['stock'];
```

```
    $sql="insert into coches values(null,'$modelo','$marca',$precio,$stock)";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Registro insertado correctamente";
  }
catch(PDOException $e)
  {
  echo $sql . "<br>" . $e->getMessage();
  }

$conn = null;
?>
```

EJERCICIOS

Base de datos concesionario (Tabla coches)

Nota: Guarda en ficheros independientes código que se repite en una carpeta **/includes** (header, nav, footer y conexionBD). Incluyelos dentro de u código utilizando **require_once**

- ✓ **Ejercicio 1.** Crea una aplicación web con una maquetación simple Header, nav horizontal y footer para todas las páginas. En el menú de navegación
  - **Principal (**Donde se muestren el listado de coches en una tabla con estilos)
  - **Insertar coche** (Formulario con los campos necesarios). Si se inserta correctamente la página se redirige a la pantalla principal y si tiene algún error volverá al formulario.
  - Vamos a incluir al lado de cada registro de coche de la página Principal una imagen de lápiz para editar y otra imagen para borrar. Cada imagen tendrá un enlace con el id del registro correspondiente. Utilizaremos GET para la recogida de datos por la URL

    Ejemplo: borrarcoche.php?id=5

Utiliza restricciones de HTML (required y campo number)

La recogida de datos (TIPO STRING) del formulario se realizará utilizando la función mysql_real_escape_String y Trim. Con estas funciones conseguimos que ningún hacker altere la ejecución del SQL.

```
$nombre = isset($_POST['nombre']) ? mysqli_real_escape_string($conexion, trim ($_POST['nombre'])) : false;

$numero= (int) $_POST['numero'];
```

Cuando se hayan insertado los datos correctamente en los formularios redirige a la página principal del listado de coches.

header("Location: pagina.php");

Si se ha producido algún error (Ej campo not null vacio) en los formularios guarda los mensajes de error en un array asociativo **$errores** y una vez comprobados todos los campos redirige al mismo formulario (Todavía no mostraremos el mensaje de error).

- ✓ **Ejercicio** 2.
  - Crea una tabla de usuario dentro de la BD concesionario con los campos (email, password, nombre, apellidos, edad , dirección) introduce los tipos necesarios.
  - Incluye un doble formulario en la pantalla (Index.php) para un control de acceso a nuestra aplicación web. Puedes utilizar el mismo header y footer (Sin el menú de navegación-nav)
    - Un formulario de control de acceso email y contraseña para entrar en la aplicación web. Si se loguea correctamente se redirige a la página principal (listado de coches).
    - Segundo para registrarte email, contraseña, nombre, apellidos, edad, dirección. Guarda el hash del password en la BD junto con los demás datos.
    - **Nota:** Se recomienda que cada formulario vaya a un php independiente para menor complejidad.

Realizar el hash del password

```
$password_segura = password_hash($password, PASSWORD_BCRYPT, ['cost'=>4]);
```

Comprueba que la contraseña coincide con su hash

```
$verify = password_verify($password, $usuario['password']);
```

    - Utiliza la función mysqli_real_escape_string para recoger los datos string como vimos en el apartado anterior y crea también también un array asociativo $errores (Comprueba que los campos no estén vacios, email sea válido y guarda también si hay algún problema en el insert).

Nota: Una vez acabe el registro se redirige a la misma página los errores los mostraremos cuando lleguemos el apartado de sesiones.

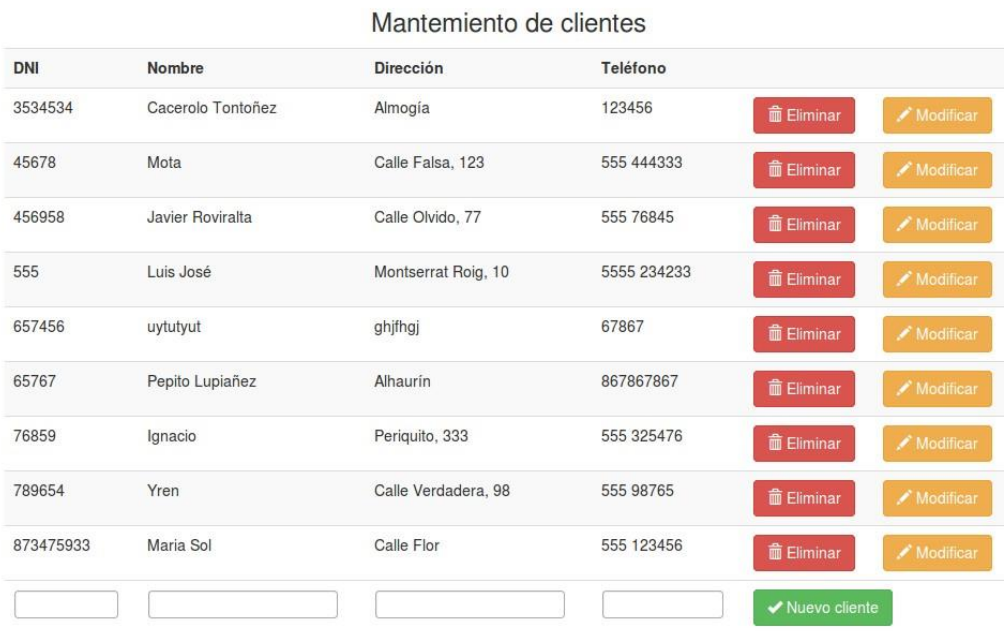
```
$email = isset($_POST['email']) ? mysqli_real_escape_string($db, trim($_POST['email'])) : false;
if(empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)){
        $errores['email'] = "El email no es válido";
}
//Una vez comprobado todos los campos
if(count($errores) == 0){…….}
```

- ✓ **Ejercicio 3:** Crea una nueva opción en el menú llamadas Listado de clientes. Será un formulario con una lista de selección de todos los clientes y un botón para mostrar una tabla con Modelo y marca de la tabla coches junto con cantidad y fecha de tabla encargos. Utiliza la misma página para el formulario y mostrar los resultados por php.
  Nota: Utiliza inner join para unir las 3 tablas (clientes, encargos y coches) y con el where selecciona el cliente del formulario por la id

```
SELECT T1.Col1, T1.Col2, T1.Col3, T2.Col7 FROM Tabla1 T1 INNER JOIN Tabla2 T2 ON T1.Col1 = T2.Col1
```

Base de datos banco

- ✓ **Ejercicio 4**: Crea una aplicación web que permita hacer listado, alta, baja y modificación sobre la tabla cliente de la base de datos banco.
  - Para realizar el listado bastará un SELECT, tal y como se ha visto en los ejemplos.
  - El alta se realizará mediante un formulario donde se especificarán todos los campos del nuevo registro. Luego esos datos se enviarán a una página que ejecutará INSERT.
  - Para realizar una baja, se mostrará un botón que ejecutará DELETE.
  - La modificación se realiza mediante UPDATE.

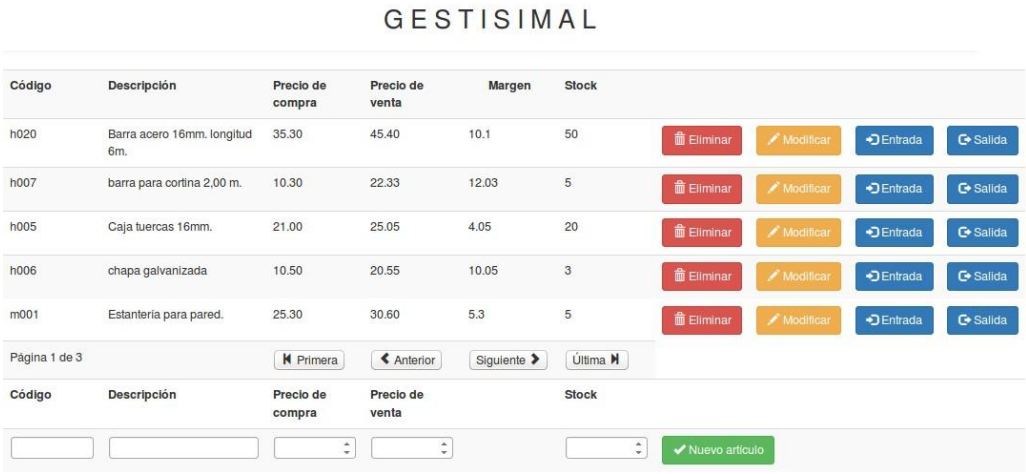


- ✓ **Ejercicio 5**: Modifica el programa anterior añadiendo las siguientes mejoras:
  - El listado debe aparecer paginado en caso de que haya muchos clientes.
  - Al hacer un alta, se debe comprobar que no exista ningún cliente con el DNI introducido en el formulario.
  - La opción de borrado debe pedir confirmación.

- Cuando se realice la modificación de los datos de un cliente, los campos que no se han cambiado deberán permanecer inalterados en la base de datos.

Base de datos gestimal

✓ **Ejercicio 6**: Crea el programa GESTISIMAL (GESTIón SIMplifcada de Almacén) para llevar el control de los artículos de un almacén. De cada artículo se debe saber el código, la descripción, el precio de compra, el precio de venta y el stock (número de unidades). La entrada y salida de mercancía supone respectivamente el incremento y decremento de stock de un determinado artículo. Hay que controlar que no se pueda sacar más mercancía de la que hay en el almacén. El programa debe tener, al menos, las siguientes funcionalidades: listado, alta, baja, modificación, entrada de mercancía y salida de mercancía.



✓ **Ejercicio 7**: Modifica el programa anterior añadiendo las siguientes mejoras:

- Comprueba la existencia del código en el alta, la baja y la modificación de artículos para evitar errores.
- Cambia la opción "Salida de stock" por "Venta". Esta nueva opción permitirá hacer una venta de varios artículos y emitir la factura correspondiente. Se debe preguntar por los códigos y las cantidades de cada artículo que se quiere comprar. Aplica un 21% de IVA.

3 Sesiones y cookies

Sesiones

Las sesiones se utilizan en PHP para guardar información en la memoria RAM. Esta información no se pierde si el usuario salta de una página a otra.

Hemos visto anteriormente cómo enviar datos entre dos páginas mediante un formulario, ahora imagina que visitas varias veces la misma página, que saltas a otra, que vuelves de nuevo a la primera; sería muy engorroso estar mandando datos constantemente mediante formularios entre unas páginas y otras. Mediante las sesiones se puede conservar o modificar la información que nos interese independientemente de la/s página/s que se vayan visitando. El valor que se almacena en la memoria está disponible mientras no se cierre la sesión.

Un uso típico de una sesión es el carrito de la compra de una tienda on-line. Podemos visitar todas las páginas que queramos de la tienda e ir añadiendo o quitando productos del carrito gracias a que esta información se graba en una sesión.

La sesión comienza con la función **session_start()** y debe colocarse siempre al principio, antes de mostrar cualquier cosa en el documento HTML.

Para cerrar la sesión se utiliza **session_detroy();**

El ejemplo más sencillo del uso de sesiones es un contador de visitas a una página.

```
<?php
session_start(); // Inicio de  sesión
if(isset($_SESSION['visitas'])) {
        $_SESSION['visitas']++;
} else {
        $_SESSION['visitas'] = 1;
}
?>
<!DOCTYPE  html>
<html>
<head>
<meta  charset="UTF-8">
</head>
<body>
<?php
echo   "Visitas: ".$_SESSION['visitas'];
?>
</body>
</html>
```

Nota: Todos los ficheros PHP empezar con sesión_start(); para poder utilizar las variables de sesión.

Cookies

Una ***cookie (galleta en inglés)*** es un fichero que se graba en el ordenador del propio usuario, no en el servidor. Permite guardar información de tal forma que no es necesario enviarla mediante formularios al pasar de una página a otra. Una cookie es algo parecido a una sesión aunque, a diferencia de esta última, la cookie se graba en el disco duro del ordenador.

Las cookies se crean con la función **setcookie()** que debe estar situada al comienzo de la página, antes que cualquier etiqueta HTML. El formato de esta función es el siguiente:

setcookie(nombre, valor, momento de expiración)

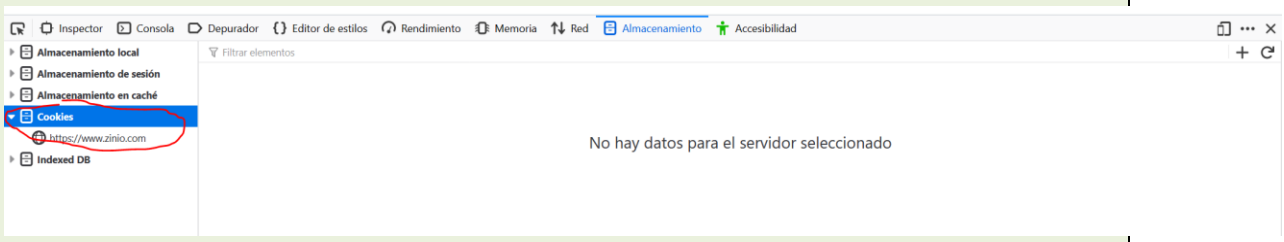
Ejemplo: setcookie("usuario", "Luis", time() + 7*24*60*60)

El momento de expiración es aquel en el que la cookie se elimina y se expresa en segundos. Normalmente se utiliza la función **time()** junto con el número de segundos que queremos que dure la cookie. Por ejemplo, si queremos crear una cookie de nombre usuario, inicializada a Luis y que dure una semana, escribiríamos lo siguiente:

Si no se especifica el momento de expiración, la cookie durará hasta que se cierre el navegador.

Para recuperar el valor de cualquier cookie se utiliza el array **$_COOKIE['usuario']**.

A la hora de depurar un programa, es muy útil mostrar todas las cookies. Podemos hacer esto con `var_dump($_COOKIE)` o a través de inspector del navegador




```
<?php
// Si se envían datos desde el  formulario de  actores,
// se actualizan las cookies

if (isset($_POST["actriz"])) {
            $actriz =  $_POST["actriz"];
            $actor =  $_POST["actor"];
           setcookie("actriz",  $actriz,  time() +  3*24*3600);
           setcookie("actor",  $actor,  time() +  3*24*3600);
} else if (isset($_COOKIE["actriz"])) {
           $actriz =  $_COOKIE["actriz"];
           $actor =  $_COOKIE["actor"];
}
```

```
// Borrado de  cookies y variables


if (isset($_POST["borraCookies"])) {
 setcookie("actriz", NULL, -1);
 setcookie("actor", NULL, -1);
unset($actriz);
unset($actor);
}
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
<?php
if (!isset($actriz)) {
echo "No has elegido todavía a tus actores favoritos.<br/>";
echo "Utiliza el siguiente formulario para hacerlo.<br/>";
} else {
echo "<h2>Actriz favorita: ".$actriz."</h2>";


echo "<h2>Actor favorito: ".$actor."</h2>";
echo "Introduce nuevos nombres si quieres cambiar tus preferencias.<br/>";
}
?>
<form action="#" method="post">
Actriz: <input type="text" name ="actriz"><br> Actor: <input type="text" name
="actor"><br>
<input type="submit" value="Aceptar">
</form>
<hr/>

<form action="#" method="post">
<input type="hidden" name="borraCookies" value="si">
<input type="submit" value="Borrar cookies">
</form>
</body>
</html>
```

Ejercicios

- ✓ **Ejercicio 1 (SESIONES):** Vamos a ampliar nuestro ejercicio del coche con variables de sesión, utilizaremos una variable de sesión para **guardar el usuario** que se logué correctamente y que aparezca siempre el nombre de usuario en el header o en el nav. Otra de las variables de sesión que vamos usar es la de **errores** para mostrarlo en el formulario.

- Primero es iniciar sesión en todos los ficheros y aprovechamos para comprobar si un usuario ha intentado entrar a nuestra aplicación web sin loguearse. crearemos un fichero en **/includes** que sea **redireccion.php** que introduciremos al principio de cada página.

```
<?php
if(!isset($_SESSION)){session_start();}
if(!isset($_SESSION['usuario'])){header("Location: index.php");}
} ?>
```

- Crear en el menú navegación un enlace de **cerrar sesión** que te redirija al **index.php**. Dentro del index.php tendrás que comprobar si existe $_SESSION y borrar todas las variables **($_SESSION=array();)**y destruir la sesión (**destroy_session();**).
- **Control de errores:** Crea un fichero php **include/helpers.php** y realiza una función (**mostrarError**(**$errores):string**). se ejecutará delante del formulario devolviendo un string con la lista de errores si existe la variable (Cada error con un div con fondo rojo). Una vez que muestre los errores tendrás que liberar la $_SESSION[‘errores’]. En el formulario registro si no existe la variable $_SESSION[‘errores’] muestra una línea de registro insertado correctamente.
- **Para mayor seguridad Introduce también un if en el nav que compruebe que existe la sesión de usuario antes de mostrar las opciones de la lista del menú**

✓ **Ejercicio 2 (COOKIES):** Crear dentro del formulario de Login del ejercicio coches un **campo recordar (Checkbox)** donde guardes el usuario y contraseña en una cookie para que se muestre las próxima vez que se cargue el formulario

Ejercicios de ampliación

✓ **Ejercicio 2 (Sesiones):** Crea una tienda on-line sencilla con un catálogo de productos y un carrito de la compra. Un catálogo de cuatro o cinco productos será suficiente. De cada producto se debe conocer al menos la descripción y el precio. Todos los productos deben tener una imagen que los identifique. Al lado de cada producto del catálogo deberá aparecer un botón Comprar que permita añadirlo al carrito. Si el usuario hace clic en el botón Comprar de un producto que ya estaba en el carrito, se deberá incrementar el número de unidades de dicho producto. Para cada producto que aparece en el carrito, habrá un botón Eliminar por si el usuario se arrepiente y quiere quitar un producto concreto del carrito de la compra. A continuación se muestra una captura de pantalla de una posible solución.

- ✓ **Ejercicio 3:** Amplía el programa anterior de tal forma que se pueda ver el detalle de un producto. Para ello, cada uno de los productos del catálogo deberá tener un botón Detalle que, al ser accionado, debe llevar al usuario a la vista de detalle que contendrá una descripción exhaustiva del producto en cuestión. Se podrán añadir productos al carrito tanto desde la vista de listado como desde la vista de detalle.
- ✓ **Ejercicio 4**: Amplía el ejercicio anterior de tal forma que los productos que se pueden elegir para comprar se almacenen en cookies. La aplicación debe ofrecer, por tanto, las opciones de alta, baja y modificación de productos.