# Obteniendo datos

## Mediante PHP y MySQLi

**V 2.3**

# Tabla de contenidos

# 1. ACCEDIENDO A LA BASE DE DATOS DESDE LA WEB

Está claro que el objetivo final de toda base de datos en web es mostrar los resultados a través de una página web.

Vamos a comenzar viendo cómo podemos obtener los datos almacenados en una base de datos, evidentemente para obtener resultados visuales es necesario que tengas un "**Juego de registros de prueba**", es decir, una serie de datos que emplearemos para ir comprobando y visualizando los resultados pero que a la hora de comenzar con la explotación real no existirían puesto que entregaríamos la misma totalmente vacía.

Para realizar las operaciones emplearemos la base de datos librería y crearemos las páginas necesarias para recuperar los valores de la base de datos almacenada en determinadas tablas: **carga la base de datos que te ha facilitado el profesor y observa las tablas estructura y datos de la misma.**

**ANTES** de configurar la conexión con la base de datos debemos conocer las posibilidades existentes:

Si la base de datos está realizada con MySQL y empleamos PHP para recuperar valores podemos realizar todo el proceso desde enfoques diferentes que intentaré resumirte en la siguiente tabla de forma general:

| Versión de PHP | Versión de MySql | Método de Acceso a los datos disponibles en PHP |
|---|---|---|
| 4 e inferiores | 4 e inferiores | Procedimientos: funciones PHP para **mysql** |
| 5 y superiores | 5 y superiores | 1) Procedimientos: Funciones PHP ***mysqli*** (OBLIGATORIO EN VERSIONES DE PHP 7.1 y SUPERIORES)<br>2) Orientado a Objetos, 2 posibilidades<br>a) Con las extensiones personalizadas para MySQL<br>b) Con la extensión neutral PDO para cualquier Base de datos |

Bien, para continuar emplearemos el método quizá más sencillo independientemente de tu nivel de programación, lo haremos mediante **PROCEDIMIENTOS** empleando las **funciones MySQLi**, por lo tanto, tendrás que asegurarte de que en tu entorno de desarrollo las versiones tanto de PHP/MySQL son la 5 o superiores.

# a. Empleando PROCEDIMIENTOS

ANTES de comenzar descomprime la carpeta facilitada por el profesor libros.zip y sigue las instrucciones que se te indiquen en clase o abre el fichero INSTRUCCIONES.TXT que encontraras en su interior.

He dejado un par de carpetas y un fichero **index.php** que contiene una estructura muy sencilla que emplearemos para volcar los datos de las tablas de la BD debajo de las cabeceras de la tabla.

Veamos el orden de ejecución de las operaciones a realizar...

**Las operaciones a realizar a modo de procedimiento general son:**

1. *Configurar una conexión con la Base de datos.*
2. *Consultar la base de datos*
3. *Recuperar los resultados*
4. *Presentar el resultado al usuario.*
5. *Cerrar la conexión a la BD*
6. *Extra: Buscador - Comprobar y filtrar los datos que le llegaran al usuario.*

**1. Configurar una conexión con la Base de datos.**

Para realizar la conexión a la base de datos podemos hacerlo INSERTANDO todo el código en la cabecera del fichero en el que necesitemos dichos datos o bien, creando un fichero php SOLO con los datos de conexión y enlazando dicho fichero mediante un “include” con todos los que hagan falta de nuestro sitio, evidentemente ésta última solución es más flexible y universal, pero en este punto vamos a realizar la conexión por la primera opción, es decir, **insertando el código directamente en la cabecera del fichero, según la siguiente función:** mysqli_connect(), ésta función toma como argumentos:

- El **host**: cadena de caracteres que contiene el nombre o la dirección IP del host, que corresponde a "localhost" o 127.0.0.1 si trabaja en modo local.
- El **usuario**: cadena de caracteres que contiene el nombre de usuario para conectarse a la base de datos. Corresponde a "root" si trabajas en modo local. Cuidado, este usuario tiene todos los derechos sobre tu base de datos.
- La **contraseña**: cadena de caracteres que contiene la contraseña asociada al usuario de la BD. Por defecto se encuentra vacía.

- El **nombre de la base de datos**: cadena de caracteres opcional que contiene el nombre de la base de datos.

Esta función devuelve falso en caso de error, o un objeto mysqli que contiene el identificador de conexión en caso de éxito.

En nuestro caso la función quedaría como:

```
mysqli_connect("localhost","root","","libreria");
```

Ahora bien, para poder emplear la conexión con mayor libertad, la almacenaremos en una variable de PHP que yo voy a llamar **cnx** (de conexión), es decir.

```
$cnx = mysqli_connect("localhost","root","","libreria");
```

**Nota**: Podríamos independizar cada parámetro en variables independientes y sustituir en la función por las variables en lugar de poner los valores directamente.

Incluiremos una pequeña línea para que en el caso de que se produzca un error en la conexión nos informe de dicha situación y nos dé el número de error mediante el cual podremos investigar la causa con más precisión, ni que decir tiene que la gestión de errores puede hacerse mucho más específica incluyendo determinados mensajes según el tipo de error devuelto, pero en esencia haremos lo siguiente empleando mysqli_connect_error() y cerramos en ese momento la conexión con exit();(o die())

```
if (!$cnx) {
 printf("No se puede conectar con el servidor. Error: %s\n",
 mysqli_connect_error());
 exit();
}
```

Finalmente, el Script de PHP al completo a incluir en la primera línea de nuestro **index.php** queda:

```
<?php
 @$cnx = mysqli_connect("localhost","root","","libreria");

/*la @delante de la variable de conexión es para que los errores del
servidor no aparezcan, opción interesante cuando los gestionamos nosotros,
prueba a ponerla/borrarla y recarga la página para apreciar la diferencia
con el servidor de base de datos APAGADO*/
```

```
// Comprobar conexión

if (!$cnx) {
 printf("No se puede conectar con el servidor. Error: %s\n",
 mysqli_connect_error());
 exit();
}
?>
```

### 2 y 3 Consultar la base de datos y recuperar los resultados

Ahora vamos a proceder a realizar una consulta (SELECT), con los datos que necesitamos recuperar y almacenar las líneas de la tabla de la bd en un array que contenga los valores separados por columnas para poder "poner" cada uno en la zona que corresponde de la tabla, para ello emplearemos **mysqli_query** para construir la consulta, **mysqli_fetch_assoc** para recuperar los valores y ya que estamos **mysqli_num_rows** para contar las filas recuperadas, esto nos permitirá mostrar el típico mensaje de "Hay XX libros en la BD", la construcción global queda de ésta manera.

Este bloque tendremos que escribirlo, por ejemplo, por debajo de la etiqueta H2

```
<h2>-------- LISTADO de TODOS LOS LIBROS ACTUALES --------</h2>

<?php
       $query_resultado = "SELECT * FROM libros order by autor ASC";
       $resultado = mysqli_query($cnx, $query_resultado);
//Empleo mysqli_fetch_assoc que Recupera 1 fila de resultados como un
array asociativo, nombre de los campos de las tablas
       $row_resultado = mysqli_fetch_assoc($resultado);
      $num_resultados = mysqli_num_rows($resultado);
      echo '<p>Número de libros TOTALES:<span class="resaltado-rojo">
 '.$num_resultados.'</span></p>';
?>
```

### 4 Mostar los resultados al usuario

Ahora, aunque hemos recuperado los datos no los vemos en ningún sitio, salvo el número de registros, es decir, nos falta una construcción para recorrernos la tabla y pintar los resultados, emplearemos un bucle, en mi caso he optado por una construcción "do", en lugar de "for", para obtener siempre al menos una línea. Pega el siguiente código por debajo del comentario de la página

```
// ------------------------------------------------------
```

<! - -Empleo **do** para asegurarme de que al menos tendré un registro como resultado- ->

```
<?php do { ?>
        <td><?php echo $row_resultado['isbn']; ?></td>
        <td><?php echo $row_resultado['titulo']; ?></td>
        <td><?php echo $row_resultado['autor']; ?></td>
        <td><?php echo $row_resultado['precio']; ?></td>
      </tr>
    <?php }
 while ($row_resultado = mysqli_fetch_assoc($resultado));?>
```

**Guarda y visualiza/actualiza la página**

¿Efecto Rombos?: Veamos cómo solucionarlo.

El efecto "rombos" se produce cuando existe una descoordinación entre los distintos valores de codificación entre la base de datos y el servidor, para evitarlo hay diferentes técnicas, pero quizá la más sencilla, que no la única sea el incluir la siguiente línea ANTES de cerrar la conexión.

```
//Devolvemos el juego de caracteres UTF 8 al cliente
 mysqli_set_charset($cnx,"utf8");
```

**OJO**: Si la configuración es correcta SIN forzar el juego de caracteres no incluyas esta línea porque provocará el efecto contrario, primero se prueba sin ella, y si hace falta se incluye.

## 5 Cerrar la conexión

Bien, llegados a este punto, aunque la conexión a la base de datos se cierra de forma automática resulta más eficiente especificar el cierre de la conexión al terminar su ejecución, para ello emplearemos la siguiente línea justo antes de cerrar el script php de conexión.

```
mysqli_close($cnx);
```

*¿Fácil no?*

Evidentemente existen multitud de variaciones en relación a las operaciones anteriores que implementan más o menos información, seguridad, velocidad..., cada uno tiene su "librillo", mi opción siempre procuraré que sea la más sencilla para que lo entiendas, pero NO es la única.

## EJERCICIO

Tu solo...

1. **¿Serías capaz de mostrar una imagen de portada para cada libro almacenando en la base de datos la dirección URL de una imagen del mismo? (Haz los cambios pertinentes en la base de datos y en el código que muestra el listado)**

**apóyate para las imágenes en la casadellibro.com, por ejemplo*

**-------- LISTADO de TODOS LOS LIBROS ACTUALES --------**

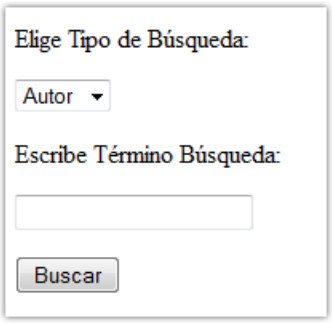
Número de libros TOTALES: 5

| Portada | ISBN | TITULO | AUTOR | PRECIO |
|---|---|---|---|---|
|  | 9788441532533 | Curso de diseño gráfico | Anna María López López | 30.00 |
|  | 9788481318593 | Un cuento de cigüeñas | Antonio Ventura | 15.50 |
|  | 9782409000560 | Powerpoint 2016: Domine las funciones avanzadas | Myriam Gris | 8.00 |
|  | 9788441537545 | windows 10 para torpes | Vicente Trigo | 15.70 |
|  | 9788441533516 | Adobe Edge animate | VV.AA. | 32.30 |

## 6 Creando un buscador

Desde el punto de vista técnico vamos a ir realizando las siguientes operaciones

Incluiremos en la página index.php un formulario con dos controles, uno de tipo select (name="tipobusqueda") con las opciones para definir el tipo de búsqueda a realizar y que serán: **autor, título** e **isbn,** además tendrá un control de tipo **texto** (name="terminobusqueda") para realizar una búsqueda por términos.




Dicha página enviará los datos por el método post a otra página php donde vamos a procesar los datos y que llamaremos **resultados.php. Créala.**

**Ábrela** con tu editor favorito y vamos a insertar el siguiente código.

Primero recuperamos los valores enviados a través del formulario en la página **resultados.php**, esto lo hacemos declarando dos variables donde almacenar el contenido enviado desde el buscador

```
<?php
  $tipobusqueda = $_POST['tipobusqueda'];
  $terminobusqueda = $_POST['terminobusqueda'];
```

Segundo vamos a filtrar los datos mediante *trim* para eliminar espacios en blanco antes o después de los caracteres que pueda introducir en la caja de texto, para recibir el valor escrito

```
trim ($terminobusqueda);
```

Ahora comprobaremos que si no se ha seleccionado un tipo de búsqueda o un término de búsqueda lo indicaremos con un error.

```
if (!$tipobusqueda || !$terminobusqueda)
  {
     echo "No has introducido todos los valores de la búsqueda.  Por favor
vuelve e inténtalo de nuevo, ambos campos son obligatorios.";
     exit();}
// ----------------------------------------------------------
```

Si sí que se han introducido datos en los controles anteriores **antes de enviar a la base de datos, PARA LO CUAL DEBERÍAMOS EMPLEAR VALIDACIONES EN EL CLIENTE MEJOR QUE EN EL SERVIDOR CON PATRONES DE HTML5 O JAVASCRIPT** debemos neutralizar el hecho de que el usuario introduzca caracteres que puedan intentar interpretar de forma literal MySql con lo que se nos generaría un error, esto es, comillas, / ...

Para ello emplearemos la función **addslashes ( )** que se encargará de anteponer la \ para que se conviertan en caracteres de escape esos posibles textos y lo aplicaremos la variable en la que se puede cometer éste error.

```
$terminobusqueda = addslashes($terminobusqueda);
```

Abrimos la conexión con la base de datos mediante la función de PHP mysqli_connect con los datos de configuración mencionados anteriormente

```
$cnx = mysqli_connect("localhost", "root", "contraseña del usuario de la
BD", "nombre_db");
```

Ahora si hay errores en la conexión un mensaje estándar

```
if (!$cnx)
  {
     echo "Error: No se ha podido conectar a la base de datos.  Por favor,
prueba de nuevo más tarde.";
     exit();
  }
```

Una vez conectados a la base de datos tendríamos que preguntar a esa base de datos (hacer la Query), y lo haremos mediante la función de PHP mysqli_query(string), que almacenaremos de nuevo en una variable, **$consulta** y recuperaremos el resultado con **$resultado**.

```
$consulta = "select * from libros where ".$tipobusqueda." like
'%".$terminobusqueda."%'";
$resultado = mysqli_query($consulta);
```

Ahora que ya tenemos el resultado de la consulta almacenado **$resultado**, emplearemos dos funciones nuevas, la primera para recuperar el número de filas que tenemos en la consulta y que se trata de **mysqli_num_rows, ( )**, es decir quedaría así, y debemos asignarlo a una nueva variable:

```
$num_resultados = mysqli_num_rows($resultado);
```

Ahora tenemos que recorrer todas las filas con un bucle for, tal que, y pondremos un echo para que nos indique el número de filas recuperadas, es decir,

```
echo "<p>Número de libros encontrados: ".$num_resultados."</p>";

 for ($i=0; $i <$num_resultados; $i++)
 {
    $row = mysqli_fetch_array($resultado);
    echo "<p><strong>".($i+1).". Título: ";
    echo stripslashes($row["titulo"]);
    echo "</strong><br>Autor: ";
    echo stripslashes($row["autor"]);
    echo "<br>ISBN: ";
    echo stripslashes($row["isbn"]);
    echo "<br>Precio: ";
    echo stripslashes($row["precio"]);
    echo "</p>";
 }
```

En el bucle empleamos la función **mysqli_fetch_array ( ),** que lo que hace es en cada ejecución del bucle toma cada fila y la devuelve en un array de cadenas con el contenido de cada fila y un valor que coinciden con el nombre de cada campo y el valor que contenga y lo almacenamos en la variable $row.

Además empleamos la función **stripslashes** para que desaparezcan en cada caso los posibles caracteres problemáticos a la hora de devolver los datos.

COMPRUEBA: Ahora sólo tienes que ejecutar la página del formulario y escribir algún dato para realizar la búsqueda en el formulario.

Una vez comprobado que ya recuperamos datos debemos realizar dos operaciones más:

**Cerrar la conexión a la Base de datos** mediante **mysqli_close($connection)**;, donde $connection es la variable que hemos empleado para habilitar la conexión, en nuestro caso $cnx, o bien cerrar simplemente la última conexión abierta con **mysqli_close( ),** las conexiones con mysqli_connect(), se cierran al finalizar la ejecución del script automáticamente.

**También:** deberíamos haber liberado el espacio en memoria ocupado por el resultado de la consulta y para ello debemos emplear **mysqli_free_result($resultado);** ***después*** de traernos los datos. Se libera la variable asociada a mysqli_query( ).

*Prueba a realizar un par de búsquedas, tanto vacías, para comprobar la seguridad, cómo con términos almacenados en la base de datos.*

## EJERCICIO

Tu solo...

1. **¿Serías capaz de mostrar la imagen de portada para cada libro almacenando en la base de datos en los resultados del buscador?**


# 2. INSERTANDO DATOS EN LA BASE DE DATOS DESDE LA WEB

Una vez que somos capaces de recuperar datos desde la base a través de una interface web, llega el momento de examinar el procedimiento básico para insertar datos en la misma.

**En realidad el proceso es muy parecido al anterior, salvo que en lugar de emplear una sentencia SELECT para solicitar datos emplearemos una sentencia INSERT para insertarlos.**

Para realizar las operaciones emplearemos la base de datos librería y crearemos una página compuesta por un formulario simple para definir los datos a insertar, con un simple formulario, y otra para el proceso de los mismos.

De nuevo, aunque la comprobación de los datos deberíamos hacerla en local con HTML5 o javascript en la página del formulario antes de enviar a la base de datos, te mostraré el proceso para recibirlos y filtrarlos con PHP

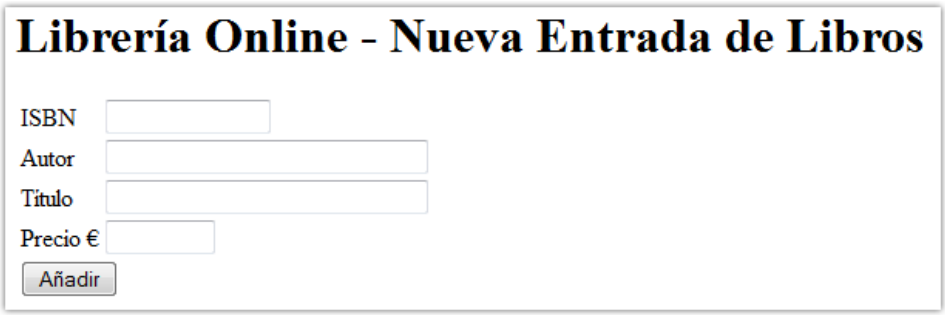
**Las operaciones a realizar a modo general son:**

1. Enviar los datos desde un formulario a la página que los procesara en la base de datos
2. Configurar una conexión con la Base de datos, en la página que recibirá los datos del formulario.

3. Comprobar y filtrar los datos que le llegaran a la base de datos. (opcional)
4. Hacer una consulta de tipo INSERT en la base de datos
5. Mostrar un mensaje al usuario con el resultado de la inserción y facilitar un enlace para volver a la página que contiene todo el listado

Para empezar, crea un enlace desde la página index a **nuevo_libro.html** con el texto "insertar libro"

1- Ahora crearemos el formulario que nos permita definir el registro a insertar, crea una nueva página **nuevo_libro.html** con el formulario que te muestro:




**Los valores de los atributos name de cada control corresponderán con su etiqueta, es decir, isbn-autor- titulo y precio**.

El formulario enviará los datos por POST a una página PHP con el nombre **insertar_libro.php**, quedaría así esencialmente:

```
<form action="insertar_libro.php" method="post">
```

Ahora vamos con el proceso y la codificación de la página **insertar_libro.php**, ábrela con tu editor favorito.

2- Abrimos la conexión a la base de datos como lo hemos realizado en otras ocasiones anteriores

**Primero recuperamos los valores de las cajas enviadas desde la página del formulario y los almacenamos en variables.**

```
<?php
   $isbn = $_POST['isbn'];
  $autor = $_POST['autor'];
  $titulo = $_POST['titulo'];
  $precio = $_POST['precio'];
```

3- Comprobamos que todos los campos a cumplimentar tienen algún dato y si no es así mostraremos un mensaje asociado, para ello recurriremos a una estructura condicional donde comprobaremos el contenido de las variables. NOTA: Este proceso deberíamos comprobarlo en el equipo local mediante validaciones HTML5 o Javascript ANTES de enviar a la Base de datos.

```
if (!$isbn || !$autor || !$titulo || !$precio)
  {
     echo "No has introducido todos los detalles necesarios. <br>"
          ."Por favor vuelve e inténtalo de nuevo.";
     exit();
  }
```

Limpiamos los datos recibidos de posibles espacios en blanco con trim

```
trim ($isbn);
trim ($autor);
trim ($titulo);
```

Preparamos los datos a enviar a la BD para evitar caracteres que den posibilidades de error, es decir, volveremos a emplear la función **addslashes** ( ) para los campos de texto y una nueva **doubleval ($variable)** (ésta función es un alias de floatval), **para los números decimales**, ésta función convierte los números a flotantes y así prevenimos la introducción errónea o con símbolos monetarios. En el campo precio

```
$isbn = addslashes($isbn);
$autor = addslashes($autor);
$titulo = addslashes($titulo);
$precio = doubleval($precio);
```

4- Ahora vamos con la parte de la inserción

Creamos dos variables, una para almacenar el código SQL correspondiente al INSERT a Realizar en la tabla de la base de datos y otra para recoger el resultado obtenido a través de **mysqli_query()**

```
$query = "INSERT INTO TABLA (campo1, campo2, campo n) VALUES
('variable1/valor1', 'variable2/valor2', 'variablen/valorn')";

$result = mysqli_query($cnx, $query);
```

**Nota:** Observa la consulta que hacemos, evidente, tipo **insert**.

Ahora escribiremos el código para recuperar el estado de la inserción y mostraremos un mensaje que indique el final del proceso y cerraremos el script.

```
if ($result)
      echo  mysqli_affected_rows($cnx)." libro introducido en la base de
datos.";
?>
```

Cuando realicemos cambios en la base de datos, inserción (INSERT), eliminación (DELETE) o actualización (UPDATE), necesitamos emplear la función mysqli_affected_rows ($variable_de_conexion) para que nos indique cuantas filas se han visto afectadas en el proceso, o por lo menos es lo correcto, porque técnicamente podríamos prescindir de ésta funcionalidad y realizar la inserción sin más.

**Para comprobar el proceso, inserta un nuevo libro a través del formulario y luego búscalo mediante la página con el buscador que realizamos anteriormente**.

Prueba con estos datos:

- ISBN: 09-123456
- Autor: Tu nombre y Apellido
- Título: Aprendiendo PHP y MySQL
- Precio: 40€

Deberías ponerte una serie de enlaces para conectar las páginas de resultado de la inserción y el buscador de la librería.

## EJERCICIO

Tu solo...

1. **¿Serías capaz de incluir un campo imagen en el formulario de inserción para almacenar la imagen elegida desde una url?**


2. **En el Caso de que el campo PORTADA (inserción de la URL de la imagen), NO SEA OBLIGATORIO ¿serías capaz de cambiar el desarrollo para?**

    a. Primero en la base de datos para que en el campo de la imagen esté definida una imagen por defecto en el caso de no almacenarse ninguna.

    b. Cambiar el código en la página **insertar_libro_mysqli.php** para que en el caso de que no se inserte ningún libro en el formulario lea la imagen por defecto establecida en la base de datos.

# 3. CONEXIÓN GLOBAL PARA TODAS LAS PÁGINAS

Ahora bien, con la estructura vista hasta el momento tenemos que configurar la conexión con la BD en cada página así que lo práctico sería extraer dicha conexión a un fichero externo, por ejemplo, "**connection.php**" donde recojamos todos los datos y después emplear una llamada a dicho fichero mediante require ( ) o include ( ) en sus diferentes variantes.

Es decir una estructura de éste estilo:

```
----------------- connection.php -----------------
<?php
$cnx = mysqli_connect("localhost","root","","libreria");

// Comprobar conexión
if (mysqli_connect_errno())
  {
  echo "Conexión denegada, el Servidor de Base de datos que solicitas NO
EXISTE" . mysqli_connect_error();
  }

  //Devolvemos el juego de caracteres UTF 8 al cliente

  mysqli_set_charset($cnx,"utf8");

 //para evitar problemas de codificación de vuelta a la página web
 //podemos añadir esta línea al fichero de conexiones
    //mysql_set_charset ($cnx,"utf8"); PERO

 //Si codifica de forma correcta y se pone esta línea se produce el efecto
contrario //se decodifican los caracteres y tenemos el "efecto rombos".

     //Si el problema está cuando hacemos consultas a la Base de datos lo
más //efectivo suele ser añadir la siguiente línea a la conexión.

//mysqli_query($cnx,"SET NAMES 'utf8'");
  //ESTE TIPO DE PROBLEMAS SUELE PRODUCIRSE CUANDO EN EL SERVIDOR O EN LA
BASE DE //DATOS NO ESTA DEFINIDO EL JUEGO DECARACTERES CORRECTO, LO
NORMAL, ES QUE NO //TENGAMOS QUE INCLUIR ÉSTE PROCEDIMIENTO
    ?>
// ----------------------------------------------------
```

Y por lo tanto en las páginas donde queramos realizar la conexión llamaríamos a este fichero mediante

```
<?php require_once('connection.php'); ?> O Bién

<?php include_once('connection.php'); ?>
```

Y nos evitamos repetir el código correspondiente a la conexión en la cabecera de todas las páginas que realicen operaciones en la base de datos.

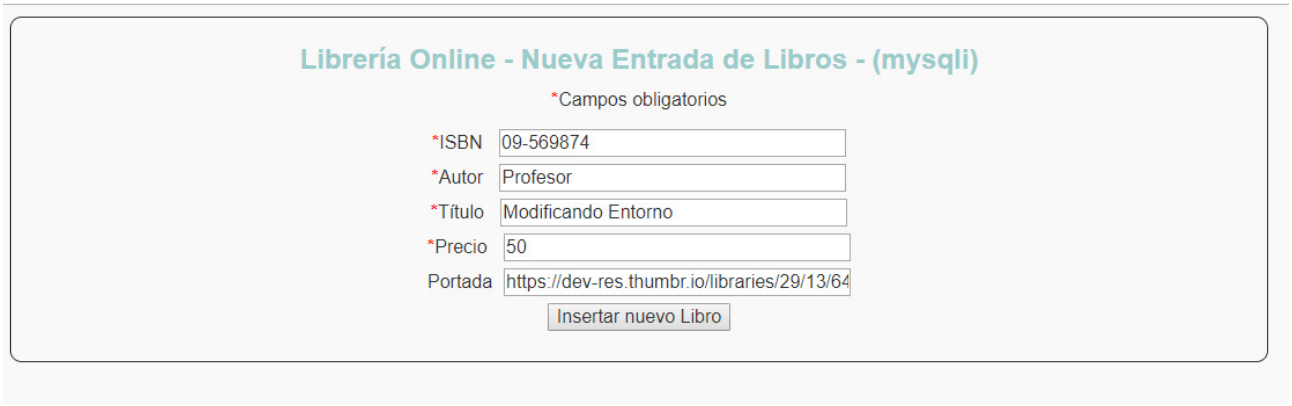
**Nota:** aunque **mysqli_connect** no establece una conexión permanente con la bd al terminar la ejecución del script la conexión se cierra sola, pero resulta interesante cerrar al finalizar el script mediante mysqli**_close($conexion_db);**

## Ejercicio: Sobre la base de datos Librería

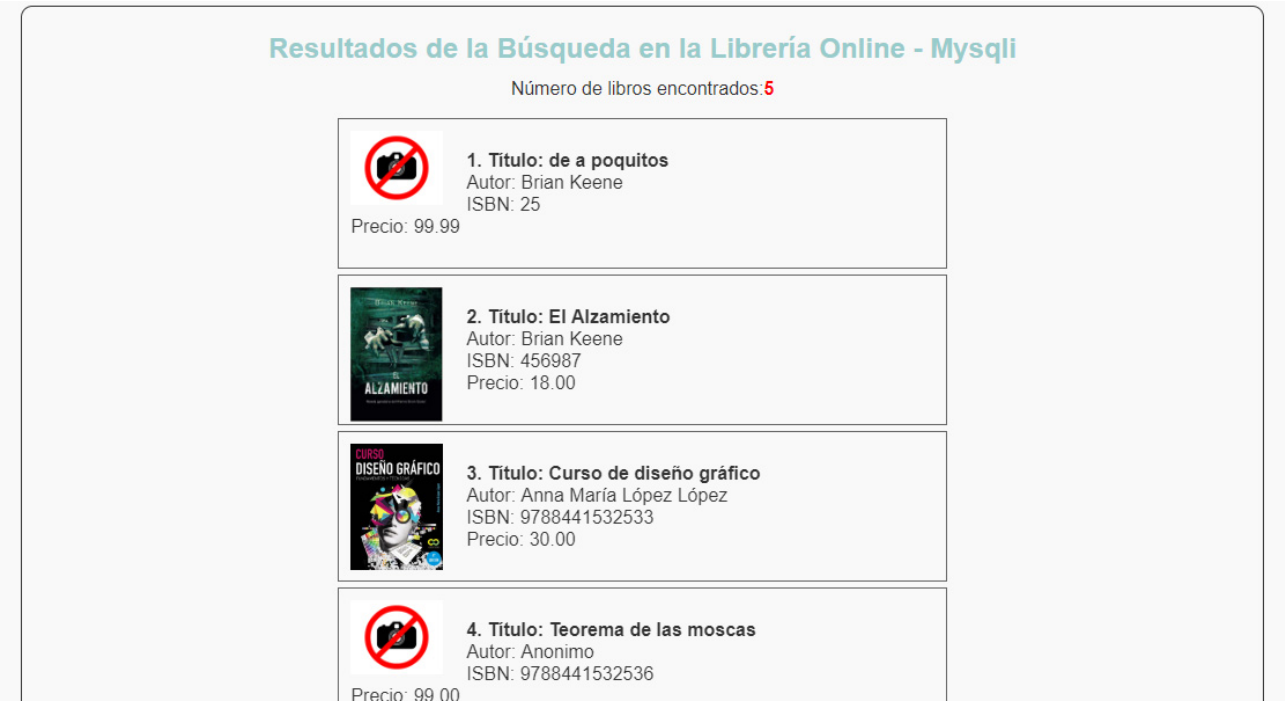
1. Crea el fichero que contenga todas las conexiones a la BD.
2. Incluye dicho fichero en las páginas que necesites conectar con la Base de datos y elimina el código anterior que realizaba ésta conexión)

**Imágenes finales del proceso**

**Inserción con URL de la portada de un libro**



**Resultados Buscador con la caratula del libro o su imagen por defecto**




```
//CAMBIAMOS DE BASE DE DATOS
```

# 4. ACTUALIZANDO DATOS

### Preparación del entorno de desarrollo

Antes de que continúe con las dos operaciones pendientes, la **actualización** y la **eliminación** de registros, voy a facilitarte los ficheros básicos para proceder, emplearemos una nueva base de datos "**directorioempresas**" a través de un fichero *sql que debes importar mediante phpMyAdmin, y que ya tiene registros.

También te facilito otro fichero sql solo con más registros, que de momento no hace falta, es sólo por si eliminas más de la cuenta que no tengas que teclear nuevos, bastará con importar el segundo fichero en la base de datos.

a) DB y Registros Extra: Descárgatelo desde aquí e impórtalos.

Te facilito también una **carpeta** con las páginas básicas (Framework), una que genera un listado actual de los registros y el fichero de la conexión y ya he incluido una hoja de estilos muy simple para darle algo de forma, así como una página con la base para editar.

b) Descárgatelo desde aquí, descomprime y pasa la carpeta al directo www del servidor y crea un nuevo proyecto o sitio a la carpeta según tu editor

- Comprueba que puedes observar el listado cargando la página **listado.php**

- Observa el código de la página **conexion.php, editar.php**

## E J E R C I C I O

**Sobre la base de datos directorioempresas**

1. Podrás observar en las páginas facilitadas que los procedimientos se basan en **mysql**, la primera operación a realizar es sustituir esos métodos por los equivalentes a **mysqli**: **HAZLO**.

2. Incluye dicho fichero en las páginas que necesites conectar con la Base de datos y elimina el código anterior que realizaba esta función

3. REALIZA EL PROCEDIMIENTO PARA INSERTAR NUEVAS EMPRESAS COMO SE HA REALIZADO EN EL EJEMPLO DE LA LIBRERÍA

    a. Verifica que puedes insertar nuevas empresas y que al hacerlo se redirecciona al listado para observar el proceso.

**ACTUALIZANDO DATOS EN LA BASE DE DATOS DESDE LA WEB**

Para poder realizar las modificaciones sobre un registro tenemos un procedimiento que consta de los siguientes pasos y operaciones.

1. Habilitar un enlace dinámico desde la página donde queramos iniciar el proceso para conectar con la que REALIZARA el proceso, en dicho enlace ira identificado el registro de forma inequívoca (parámetro)

2. Crear un formulario en la página que recibirá el identificador pasado desde la otra página que mostrará en sus cajas los datos correspondientes al registro a editar.

3. Una vez realizadas las modificaciones enviar los nuevos valores a la base de datos mediante el formulario.

**PROCEDIMIENTO**

Aunque el enlace y la página que lo recibe podemos crearlas en el orden que queramos para obtener confirmaciones visuales resulta práctico crear primero la página que recogerá los datos a editar, en este caso te la facilito yo con el formulario creado y parte del código necesario, abre el fichero **editar.php** con el editor No con el servidor, cómo te digo falta código y te mostrará al "predicador"

**Primero:** deberemos crear un enlace desde la página que contenga los registros a la página que utilizaremos para realizar los cambios. Ahora bien, dicho enlace no puede ser un

enlace ordinario dado que realizaremos el cambio **únicamente** sobre la empresa seleccionada por lo que tendremos que crear un enlace de tipo dinámico **para pasar un valor que lo identifique de manera inequívoca en la tabla**, y que normalmente será el **id**, de ahí, el definirlo como campo autonumérico evitando que se repita el valor, veamos cómo, tomaremos cómo referencia la página que tiene el listado.

1.: Crear la página que recogerálos datos enviados desde el listado **editar.php, que ya te la facilito yo.**

2.: En la página del listado, **listado.php** vamos a convertir en enlace dinámico el texto editar situado al final de cada registro listado.

a.: Editamos dicho enlace para que cuando conectemos con la página de edición lleve incluido un identificador exclusivo de cuál es el que vamos a modificar, observa.

```
<a href="editar.php?id=<?php echo $rsEmp['id']; ?>">Editar</a>
```

Es decir creamos un parámetro *id* unido a la url mediante ? y lo igualamos ("lleva") el valor recuperado mediante la consulta en el campo indicado.

Si bien el nombre del parámetro puede ser cualquiera el registro identificador NO, por lo que podríamos tener algo así:

```
<a href="editar.php?pepe=<?php echo $rsEmp['id']; ?>">Editar</a>

---ESTO ES LO QUE SE CONSIDERA UN ENLACE DINÁMICO---
```

3.: **Comprobemos** si "pasan bien" los datos a la página que los recogerá:

a.: Para ello ejecuta la página del listado y pasa el cursor, no pulses, sobre los diferentes enlaces y observa el valor que pasarías en la barra de estado del navegador.

4.: Vamos con la página que recibirá dichos datos.

a.: Si has pulsado cualquier enlace en la página anterior, y si no, ahora es el momento, observaras que el formulario no muestra nada porque realmente no está recogiendo nada de lo que le estamos enviando en el parámetro, necesitamos una consulta en ésta página que los recoja.

b.: Para ello en la página **editar.php** crearemos una consulta que los recoja, pero no todos los id de empresa sino sólo aquel cuyo valor corresponda al parámetro enviado.

c.: Vamos a incluir las siguientes líneas de código a continuación de los requiere

//Abrimos una variable para recoger el valor enviado en el enlace (Método Get)

```
$idempresa = $_GET["id"];
//obtenemos los datos del registro seleccionado, el que se nos pasa por
get que es su id, para que las cajas tengan contenido
$sql = "SELECT * FROM empresa WHERE id = ".$idempresa;
$queEmp = mysqli_query($conexion, $sql);
$rsEmp = mysqli_fetch_assoc($queEmp);
$total = mysqli_num_rows($queEmp);
//Metemos una pequeña condicion por si se pulsa el botón de
Guardar/Actualizar sin hacer ningun cambio ir de nuevo al listado
if ($total == 0) {
     header("Location: listado.php");
     exit;}
```

Guardamos la página y partiendo **desde listado** y pulsando en uno de los enlaces, preferiblemente que no sea el primero, de EDITAR, debemos recuperar en la página editar.php que los datos que se recogen en las cajas son los que corresponden a ese registro

**HAZLO**

```
//Compruebo recuperación en las cajas de formulario, en un registro QUE NO SEA
EL PRIMERO
```

d.: Una vez que ya recuperamos el dato correcto y vamos a grabarlo en la base de datos empleando una página intermedia, que aunque podríamos realizar todo el proceso en la misma página, no es elmétodo aconsejado para poder implementar en ésta medidas de control de los valores antes de enviarlos a la base de datos.

e.: Crea la página **editar_procesa.php** y escribe el siguiente código en su interior

```
<?php require_once("conexion.php");
//Recogemos datos enviados desde editar.php
       $idempresa = $_POST["id"];
       $nombre = $_POST["nombre"];
       $direccion = $_POST["direccion"];
       $telefono = $_POST["telefono"];
//Creamos la sentencia Sql para realizar la actualización
       $sql = "UPDATE empresa SET ";
       $sql.= "nombre='".$nombre."', direccion='".$direccion."',
telefono='".$telefono."' ";
       $sql.= "WHERE id=".$idempresa;
// ----------------------------------------------------
```

```
        mysqli_query($conexion, $sql); //Sigue...
    //Redireccionamos
        header("Location: listado.php");
    ?>
```

f.: Además de realizar la consulta de actualización es necesario pasar en un **campo oculto del formulario** el **valor por el que se guiará el proceso para encontrar la fila en la tabla adecuada**, es decir el id. **Asegúrate en el formulario editar.php de que existe la siguiente expresión**.

```
<input type="hidden" id="id" name="id" value="<?php echo $rsEmp["id"]; ?>" />
  Y también de que el formulario en el action, apunta a la pagina
  editar_procesa.php
```

Cómo ves tampoco es excesivamente complicado, ahora comprueba que puedes editar los registros de la Base de datos.

# 5. ELIMINANDO DATOS EN LA BASE DE DATOS DESDE LA WEB

Para eliminar datos desde la web en una base de datos seguiremos un procedimiento muy parecido, únicamente variará en la solicitud de confirmación de la operación, que si bien no es absolutamente necesaria, debemos incluirla para evitar borrados no deseados, y que te la mostraré con una pequeña función de Javascript, aunque podríamos pasar los datos a una página intermedia que mostrara el registro a eliminar y ahí pediríamos confirmación.

**La lógica del proceso es:**

1. Habilitar un **enlace dinámico** desde la página donde queramos iniciar el proceso para conectar con la que REALIZARA el proceso, en dicho enlace ira identificado el registro de forma inequívoca

2. Mostrar un mensaje de alerta avisando del borrado de registro, esto podemos hacerlo:

    a. Mostrar un mensaje de alerta, normalmente con Javascript.

    b. Crear un formulario en una página que recibirá el identificador pasado desde la otra y que mostrará en sus cajas los datos correspondientes al registro a eliminar, dicho formulario no será editable y pedirá la confirmación de eliminación.

3. Una vez aceptada la confirmación se ejecuta la consulta de eliminación.

**PROCEDIMIENTO**

1. Lo primero, es crear el enlace en la página del listado, así que, crea un texto "eliminar" y pasa el parámetro a una supuesta página **eliminar.php**, no pasa nada si no existe, la vamos a crear…

    a. Ahora bien ese parámetro debemos facilitárselo previamente a una función de javascript que es la que mostrará el aviso de confirmación, y en caso afirmativo, continuará ahasta la página de eliminación.

    b. Crea el siguiente enlace en el **listado.php** si es que no existe ya:

```
<a href="#" onclick="delEmpresa(<?php echo $rsEmp['id']; ?>);">Eliminar</a>
```

Observa, estoy pasando el parámetro id a una función delEmpresa de javascript, que de momento no tenemos definida así que,

2. Por detrás del cierre de la tabla vamos a escribir dicha función, que mostrará el aviso y "pasará" la acción a realizar a la página de eliminación.

```
<script type="text/javascript">
function delEmpresa(id) {
 if (window.confirm("Aviso:\nDesea eliminar el registro seleccionado?")){
       window.location = "eliminar.php?id="+id;
 }
}
```

3. Ahora creamos la página que eliminará y que llamaremos **eliminar.php** con el siguiente código en su interior.

```
<?php
require_once("conexion.php");

$idempresa = $_GET["id"];

 $sql = "DELETE FROM empresa WHERE id=".$idempresa;
 mysqli_query($conexion, $sql);
 header("Location: listado.php");
?>
```

**Nota:** Si hubiéramos elegido mostrar el registro a eliminar en una página de confirmación "intermedia", deberíamos haber procedido cómo en el caso de la actualización de registros explicado anteriormente.

Es decir, desde el enlace dinámico a una página tipo **confirma_eliminación.php** donde en un las capas o celdas de la tabla recuperaríamos los valores del registro a eliminar y desde esa, con un botón o enlace pasar a la página que efectúa la eliminación de forma efectiva, o bien desde esa, llamar a la función de javascript anteriormente indicada al pulsar el botón o enlace correspondiente.

## Ejercicio: Creando una base de datos.

A ver cómo se te da a ti sólo...

Tienes que diseñar una base de datos, **proyectoXXX.sql** donde al menos tengas dos tablas con la relación que consideres y que obtengas del modelo E/R que debes fabricar previamente.

Para realizar las comprobaciones necesitaras insertar un número SENSATO de registros.

Cuando consideres que la base de datos está ya diseñada estructuralmente realiza las consultas necesarias para comprobar la relación entre las tablas, estas consultas deberían ser las necesarias desde el punto de vista técnico y las que consideres desde el punto de vista práctico.

Vale, vamos con la web...

## EJERCICIO

Sobre la base de datos propietaria **Creada** crea dos páginas que permitan:

Vamos a trabajar **SOLO sobre una de tus tablas**, digamos que la principal que mostrará los registros que consideres

1. **Buscar un producto/servicio según algún campo (categoria, nombre...)**
2. **Insertar un nuevo producto/servicio en tu base de datos.**
3. **Mostrar el listado de productos/servicios de forma adecuada en una estructura controlable.**

Básicamente se trata de repetir los pasos anteriormente vistos con la librería pero con tu base de datos.

4. **Editar determinado producto/servicio**
5. **Eliminar determinado producto/servicio**
6. **Publica la BD y las páginas en tu servidor remoto**
7. **Envía la dirección URL y la BD a tu profesor para su revisión.**

# 6. SISTEMA DE AUTENTIFICACIÓN DE USUARIOS

Evidentemente hasta ahora hemos permitido a todo aquel usuario que acceda, por ejemplo, al listado del directorio de empresas anterior, que cualquiera pueda eliminar o actualizar registros y estarás conmigo que eso no va a ser lo habitual.

Lo más normal será que en lugar de mostrar los enlaces del listado correspondientes a eliminar y editar, sólo se muestren a aquel usuario que permitamos el acceso a dicha página con los enlaces pero una vez que hemos comprobado su autenticidad mediante un usuario y contraseña. Tomando como punto de partida las páginas correspondientes a la aplicación de Gestión de empresa donde ya podemos realizar todas las operaciones de administración vamos a implementar el control de acceso mediante usuario y contraseña almacenando dichos valores en un fichero.

Sin embargo, si quisieras realizar el mismo proceso basándote en una serie de usuarios almacenados en una tabla de la base de datos te dejo éste tutorial para su realización, lo cual implicaría también realizar las páginas correspondientes a la gestión de los mismos, es decir, el *Listado-Alta-Modificación y Eliminación de los mismos*.

El proceso que vamos a seguir será el siguiente:

1. Vamos a copiar la página del listado de empresa **listado.php** y la nueva copia la renombraremos como **index.php**

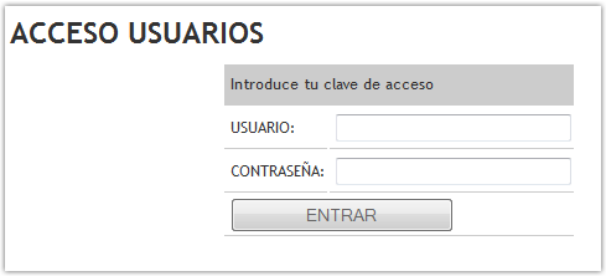
2. Elimina de **index.php** los enlaces a las operaciones de editar y actualizar, e incluye un enlace, Administrar listado, a una página nueva que llamaremos **login.php**

**Listado de Empresas Sobre la marcha**

Administrar listado

| Nombre | Dirección | Teléfono |
|---|---|---|
| AJEPER | Av. La Paz Lt.30, Sta. María de Huachipa | 917984565 |
| Courier Cristiano Jeshua | Jr. Risso 538 | 555555555 |
| Global Medica | Jr. Pablo Bermudez 192 | 4336470 |
| Global Medica | Jr. Pablo Bermudez 192 | 4336470 |
| Juegos Emperatriz | Teodoro Cárdenas 790, Sta. Beatriz | 4714603 |
| Juegos Emperatriz | Teodoro Cárdenas 790, Sta. Beatriz | 4714603 |
| Lima International School | Av. La Molina 1255, Sol de La Molina | 2525 |
| Lima International School | Av. La Molina 1255, Sol de La Molina | 4790846 |

3. Dicha página **login.php** tendrá un formulario de éste estilo




Ésta página tendrá el siguiente código **además de las cabeceras estándar habituales**:

**NOTA**: Evidentemente por facilidad de comprensión y dado que desconozco tu nivel de maquetación el código te lo doy basandome en tablas, lo suyo sería, hacer lo mismo pero basandome en capas o 

```
<form action="control.php" method="POST">
<table align="center" width="225" cellspacing="2" cellpadding="2"
border="0">
<tr>
<td colspan="2" align="center"

<?php if (@$_GET["errorusuario"]=="si"){?>
bgcolor=red><span style="color:ffffff"><b>Datos incorrectos</b></span>
<?php }else{?>
bgcolor="#cccccc">Introduce tu clave de acceso
<?php }?></td>
</tr>
<tr>
<td align="right">USUARIO:</td>
<td><input type="Text" name="usuario" size="8" maxlength="50"></td>
</tr>
<tr>
<td align="right">CONTRASEÑA:</td>
<td><input type="password" name="contrasena" size="8" maxlength="50"></td>
</tr>
<tr>
<td colspan="2" align="center"><input type="Submit" value="ENTRAR"></td>
</tr>
</table>
</form>
```

Observa, cuando introduzcamos determinados textos en las cajas dichos valores los enviamos a una página que se llama **control.php** que es la que gestionará y verificará que el usuario y la contraseña son correctos y en caso de ser así redireccionará a la página listado.php que si que tiene los enlaces para realizar modificaciones.

4. Crea una página **control.php** con el siguiente código en su interior.

```
<?php

//vemos si el usuario y contraseña es válido

if ($_POST["usuario"]=="admin" && $_POST["contrasena"]=="paso"){

//usuario y contraseña válidos
//defino una sesión y guardo datos

 session_start();

 $_SESSION["autentificado"]= "SI";

//y entonces redireccionamos a la página con el listado

 header ("Location: listado.php");

 }else {

//si no son correctos los datos de usuario y contraseña le mando otra vez
a login.php con un parámetro errorusuario=si, valor que recojo en la
página del formulario (<?php if (@$_GET["errorusuario"]=="si")) y que
provoca el cambio de color y el mensaje de aviso. La variable errorusuario,
recibida por la URL y que informa si se produjo un error al introducir
usuario y contraseña, se está recogiendo por mediación del array
asociativo $_GET, que guarda todas las variables enviadas por la URL.

 header("Location: login.php?errorusuario=si");
}
?>
```

El soporte de sesiones en PHP consiste en una manera de guardar ciertos datos a través de diferentes accesos web para no solicitarlos continuamente a través de las páginas, es decir, si la parte a la que damos acceso mediante el usuario y contraseña tiene varias páginas evitamos el pedir las credenciales en cada una de ellas si guardamos los datos en una sesión.

**session_start()** crea una sesión o reanuda la actual basada en un identificador de sesión pasado mediante una petición GET o POST, o pasado mediante una cookie.

**$_SESSION["XxX"]**, **Variables de sesión,** es un array asociativo que contiene variables de sesión disponibles para el script actual, y forman parte del procedimiento para hacer que las variables estén disponibles en múltiples páginas sin tener que pasarlas como parámetro.

A diferencia de las cookies, las variables de sesión se almacenan en el servidor y tienen un tiempo limitado de existencia. Para identificar al usuario que generó las variables de sesión, el servidor genera una clave única que es enviada al navegador y almacenada en una cookie. Luego, cada vez que el navegador solicita otra página al mismo sitio, envía esta cookie (clave única) con la cual el servidor identifica de qué navegador proviene la petición y puede rescatar de un archivo de texto las variables de sesión que se han creado.

Cuando han pasado 20 minutos sin peticiones por parte de un cliente (navegador) las variables de sesión son eliminadas automáticamente (se puede configurar el entorno de PHP para variar este tiempo).

Una variable de sesión es más segura que una cookie ya que se almacena en el servidor. La ventaja es que no tiene que estar enviándose continuamente como sucede con las cookies. Otra ventaja de emplear una variable de sesión en lugar de una cookie es que cuando el navegador del cliente está configurado para desactivar las cookies las variables de sesión, siguen funcionando (enviando la clave como parámetro en cada hipervínculo).

Como desventaja podemos decir que ocupa espacio en el servidor.

**CAPA DE SEGURIDAD**

5. Pues sí, no hemos terminado, verás es necesario añadir una "capa de seguridad".

Si ahora copias la dirección URL de acceso al listado, cierras el navegador y la pegas, te vuelves a colar sin hacer login, debemos evitarlo añadiendo un archivo a todas las páginas donde queramos evitar ese efecto **coladero**.

El módulo de seguridad, incluido al principio de cada archivo, realizará las comprobaciones oportunas y actuará permitiendo ver el archivo o denegando su visualización dependiendo de dichas comprobaciones.

Lo único que haré será recuperar la variable de sesión donde guardo si ese usuario ha sido autentificado o no. Luego se comprueba esa variable para saber si se ha autentificado el usuario o no, y realizaremos estas acciones.

a. Si no se había autentificado, redirijo al navegador a la página que tiene el formulario de autentificación. Además, salgo del script PHP, con lo que la página deja de ejecutarse y el resto no se verá. Sólo se mandará al navegador la redirección con lo que el navegador se moverá al formulario y será imposible ver nada en la página segura.

b. Si se había autentificado, no hago nada. Ni tan siquiera trato este caso, de modo que se seguiría ejecutando la página con el contenido que correspondiese.

6. Vamos a crear una nueva página que llamaremos **seguridad.php**, y en su interior irá éste código:

```
<?php
//Inicio la sesión
session_start();
//COMPRUEBA QUE EL USUARIO ESTA AUTENTIFICADO
if ($_SESSION["autentificado"] != "SI") {
//si no existe, envió a la página de autentificación
header ("Location: login.php");
//además salgo de este script
exit();
}
?>
```

7. Evidentemente en TODAS las páginas que queramos verificar la apertura de sesión debemos incluir el fichero de la manera habitual con un request o un include. Es decir, en nuestra página del **listado.php** incluye en las primeras líneas

```
require("seguridad.php");
```

**Comprobémoslo** copiando la dirección del listado, cerrando las páginas del navegador, *volviendo a abrir una página del navegador y pegando la dirección URL de acceso al listado*, si ha ido bien, debería redireccionarte a la página del login para iniciar la sesión.

**Lo normal** es incluir la misma línea en todas aquellas páginas sensibles en las que se puedan "colar" es decir, edición, eliminación e inserción de registros.

Bien, ya casi está.

## CERRANDO LA SESION

Un último paso para dejar completado nuestro micro-sistema de seguridad es la de facilitar el **cierre de sesión a petición del usuario.**

La seguridad de la aplicación se basa en la definición de unas variables de sesión que se consultan en cada página segura. Puede ocurrir que el usuario entre en la aplicación e inicie una sesión y que se marche de la aplicación segura sin cerrar la sesión, con lo que quedaría abierta para que cualquier otra persona pueda acceder a la aplicación volviendo por el historial de páginas del navegador.

**Las sesiones se finalizan solas** cuando pasa un determinado tiempo sin recibir nuevas peticiones, pero no deseamos que antes de que se finalicen se pueda acceder con ese ordenador a nuestra aplicación restringida.

Parece interesante, pues, ofrecer al visitante la opción de acabar la sesión en cualquier momento, para asegurarnos en ese caso que la sesión se ha terminado y no se podrá acceder si no es introduciendo nuevamente el usuario y contraseña correctos.

El archivo en concreto lo único que hace es terminar la sesión asociada a su acceso mediante **session_destroy();**

8. Crea una página **salir.php** con el siguiente código en su interior o similar, en realidad lo único que me interesa es el cierre de la sesión el resto podrías redireccionar otra vez al index.php, otra página…

```
<?php
session_start();
session_destroy();
?>
<html>
<head>
<title>Cerrar sesión</title>
<link href="styles.css" rel="stylesheet" type="text/css" />
</head>

<body>
<h1>Fin de la gestión administrativa
</h1>
<h2>Has cerrado la sesión</h2>
<ul>
```

```
    <li>
      <h2>Si quieres volver a administrar pulsa <a
href="login.php">aquí</a> </h2>
    </li>
    <li>
      <h2>Si quieres volver a la página principal <a
href="index.php">Aquí</a></h2>
    </li>
</ul>
</body>
</html>
```

Nos faltaría por incluir un enlace a ésta página en todas aquellas que queramos finalizar dicha sesión, es decir, en *listado* **/** *editar* **/** *insertar*, por ejemplo.

Bien, ahora sí.

**¿y es la única manera?,** pues evidentemente no, lo normal sería para empezar, tener una tabla independiente en nuestra base de datos con los usuarios que consultaríamos cuando se hiciera el login.

También podríamos mejorar el sistema definiendo un **tiempo determinado de duración de la sesión** independiente del tiempo del servidor solo tendremos que:

a. Crear una nueva sesión que guarde una fecha y hora
b. Comprobar en nuestra capa de seguridad el tiempo transcurrido entre la sesión guardada y la hora actual
c. Actualizar la sesión o destruirla según corresponda

Lo primero que debemos hacer entonces, es crear la nueva sesión y asignarle como valor, la hora actual. Esto lo haremos en el momento que el usuario ingresa al sistema con sus datos de acceso.

```
<?
//vemos si el usuario y contraseña es válido
if ($_POST["usuario"]=="admin" && $_POST["contrasena"]=="paso"){
//usuario y contraseña válidos
  session_name("loginUsuario");
//asigno un nombre a la sesión para poder guardar diferentes datos
  session_start();
// inicio la sesión
  $_SESSION["autentificado"]= "SI";
```

```
        //defino la sesión que demuestra que el usuario está autorizado
          $_SESSION["ultimoAcceso"]= date("Y-n-j H:i:s");
//defino la fecha y hora de inicio de sesión en formato aaaa-mm-dd hh:mm:ss
          header ("Location: listado.php");
                }else {
        //si no existe le mando otra vez a la portada
                header("Location: index.php?errorusuario=si");
        }
        ?>
```

El segundo paso, será comprobar el tiempo transcurrido entre la fecha guardada y la hora actual en nuestra capa de seguridad y actuar en consecuencia.

Para hacerlo, tendremos que realizar un cálculo muy sencillo:

**tiempo transcurrido = (hora actual - fecha guardada)**

Y luego, restará saber si el tiempo transcurrido es mayor, menor o igual que el tiempo de caducidad de la sesión (representado como "x"):

**si (tiempo transcurrido >= x), actúo en consecuencia a lo hallado**

Para efectuar estos cálculos utilizaremos como unidad de tiempo el segundo. En nuestro ejemplo, caducaremos la sesión, transcurridos 10 minutos de inactividad (donde: 10*60 = 600 segundos). Para efectuar estos cálculos y tomar como unidad de medida el segundo, será necesario convertir las fechas a segundos. Para ello, utilizaremos la función **strtotime**. Calcularemos el tiempo transcurrido **(tiempo transcurrido = (hora actual - fecha guardada))** de la siguiente manera:

```
    <?php
    //iniciamos la sesión
          session_name("loginUsuario");
          session_start();
    //antes de hacer los cálculos, compruebo que el usuario está logueado
    //utilizamos el mismo script que antes
          if ($_SESSION["autentificado"] != "SI") {
    //si no está logueado lo envío a la página de autentificación
          header("Location: login.php");
                } else {
    //sino, calculamos el tiempo transcurrido
                $fechaGuardada = $_SESSION["ultimoAcceso"];
                $ahora = date("Y-n-j H:i:s");
// ----------------------------------------------------
```

```
 $tiempo_transcurrido = (strtotime($ahora)-strtotime($fechaGuardada));
//comparamos el tiempo transcurrido
       if($tiempo_transcurrido >= 600) {
//si pasaron 10 minutos o más
       session_destroy(); // destruyo la sesión
 header("Location: index.php"); //envío al usuario a la pag. inicial
//sino, actualizo la fecha de la sesión
       }else {
            $_SESSION["ultimoAcceso"] = $ahora; }
}
?>
```

## 7. PAGINA DE DETALLES

Una vez que nos hemos guardado la página de administración (como index.php), eliminado las opciones administrativas y remaquetando de forma adecuada para mostrar la información de forma más amigable deberíamos a mostrar, haciendo clic sobre el elemento que seleccionemos, los detalles que pudiera tener cada registro (pelicula/empresa…), en el caso de este manual, la imagen o el enlace habilitado a tal efecto, es lo que te recomiendo.

A priori el aspecto que debería tener la web sería algo parecido a esto.

Sin embargo, considero que a estas alturas ya deberías saber hacerlo solo, así que te indico los pasos

1. Sobre la imagen o el texto adecuado, o ambos, hay que crear un enlace dinámico que apunte a la página de detalle pasándole el parámetro del id correspondiente.

2. Evidentemente hay que tener creada una página de detalle.php que reciba dicho parámetro.

3. Dicha página debe mostrar información extendida del registro recuperándola de la tabla de igual manera que en ocasiones anteriores, pero con más datos, por ejemplo, algo así.

***Al hacer clic sobre la imagen o el texto Ver detalle de la empresa se debe mostrar su página de detalle equivalente que podría ser algo así.***

# 8. MAQUETANDO LOS RESULTADOS DEL BUSCADOR

En realidad se trata de realizar la misma operación efectuada para mostrar la página de detalle, es decir.

En los resultados del buscador hay que **configurar un enlace dinámico** para que cuando se pulse en cualquiera de ellos conecte con su página de detalle, es decir, irá a **detalle.php**, **pero pasándole el ID de su registro** para que cargue sus datos.

Más o menos debería quedar algo así:

# 9. PAGINACIÓN DE RESULTADOS.

Para realizar la paginacion de resultados podemos recurrir a diferentes recursos online te indico algunos de probada funcionalidad.

1. En éste post se explica como implementar una paginación con mysqli, como precaución, tendréis que mover bloques de código y adaptar variables, entre otras, para que no se os repita el formulario por cada registro y cosas así.

   https://mimentevuela.wordpress.com/2015/12/25/paginacion-con-php-y-mysql/

2. Otra opción es un ejemplo facilitado a través de éste otro post donde se puede descargar incluso un modelo con el resultado y todo sería adaptarlo a tu desarrollo.

   https://www.baulphp.com/download/descargar-paginacion-usando-php-mysqli/

3. Algo más profesional consiste en utilizar o un framework o librería prefabricada o algún script preparado a tal efecto pero suele ser menos directa la implementación aunque cuando la tienes controlada ya será cuestión de cambiar parámetros.

**Zebra pagination.php**

4. http://daycryweb.blogspot.com/2013/01/paginacion-php-con-libreria-zebra.html

o también

**usando PHP MySqli Bootstrap**

5. https://www.baulphp.com/paginacion-usando-php-mysqli-bootstrap/

La paginación deberías implementarla en las páginas donde resulte necesario o consideres conveniente paginar, es decir, como poco en las correspondientes al **index.php** / **listado.php** / **resultados_buscador.php**

Inténtalo tu SOLO.