UD 3. Prácticas

Índice de contenidos

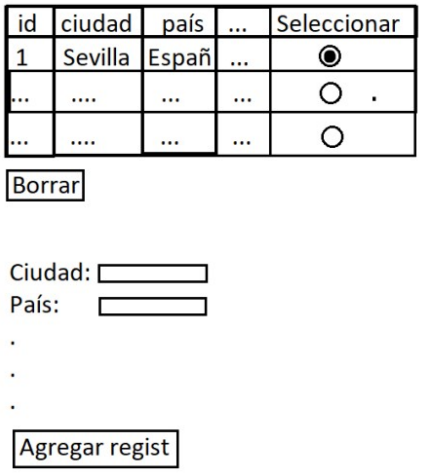
Prácticas

Práctica 1: Gestión de ciudades

Crear una aplicación para la gestión de los datos de “ciudades”. Descarga e importa la base de datos colgada en Moodle.

Considera las siguientes características:

- Todo el contenido se mostrará en una sola página.
- Deberá mostrar un listado de los campos de “ciudades” en forma de tabla. La última columna de la tabla deberá ser un selector (input de tipo radio).
- Habrá un botón de nombre "Borrar" que al pulsarlo elimine el registro que está seleccionado en el selector.
- Por último, se podrán agregar nuevos registros a “ciudades”. Para ello existirá un formulario al final de la página con los diferentes campos que componen la tabla "ciudades" y un botón llamado "Agregar registro".
- Cuando se elimine o se añada un registro, hay que mostrar el feedback. Ejemplo: mensaje de "¡Se ha eliminado el registro!".



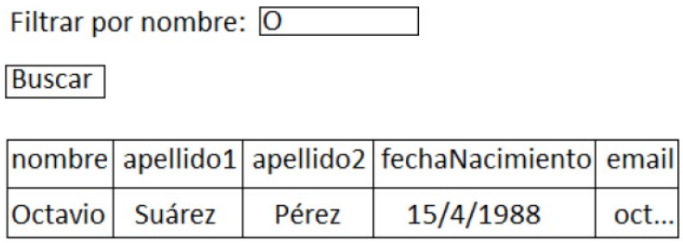
Se recomienda dividir la realización de la práctica en:

1. Crear tabla que muestre todos los registros con sus datos (sin mostrar la última columna del selector para borrar).
2. Crear formulario inferior para añadir nuevos registros a la base de datos.
3. Añadir la columna con los input para seleccionar, y el botón de Borrar.

Práctica 2: Filtrado de usuarios

Crear una aplicación para filtrar usuarios de la base de datos de “login”. Cuando se escriba en el campo “O” por ejemplo, mostrará todos los usuarios cuyo nombre empiece por O (como Octavio). En caso de que no se encuentre ningún usuario en la base de datos, no mostrará la tabla y aparecerá un mensaje “No se han encontrado resultados”.

Ejemplo visual:



Práctica 3: Aplicación de login

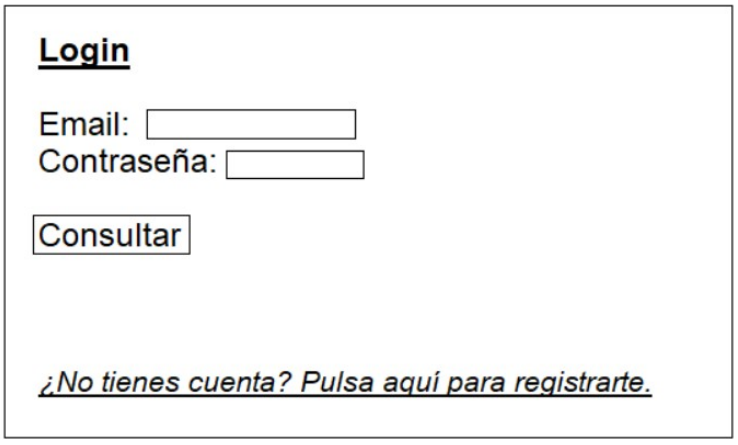
Crear una aplicación para la consulta de datos personales a partir de un email y una contraseña. La aplicación constará de dos páginas.

En la primera página el usuario podrá introducir su usuario y contraseña. Se comprobará que las credenciales son correctas. Si son correctas, se mostrarán todos sus datos personales (email, nombre, apellidos y fecha de nacimiento). Si no, se mostrará un mensaje del error.

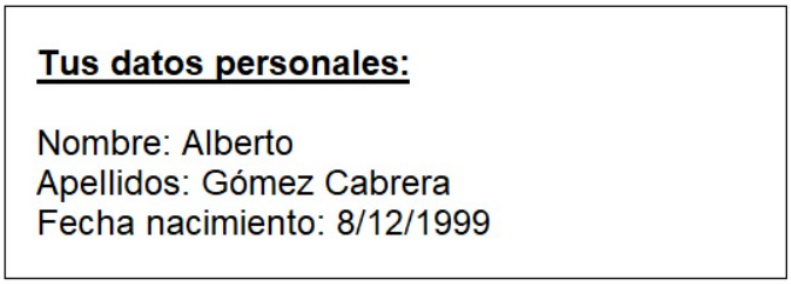
En la segunda página se permitirá que el usuario se registre en la aplicación a partir de sus datos personales.

La página 1 deberá tener un enlace para acceder a la página 2, y viceversa.

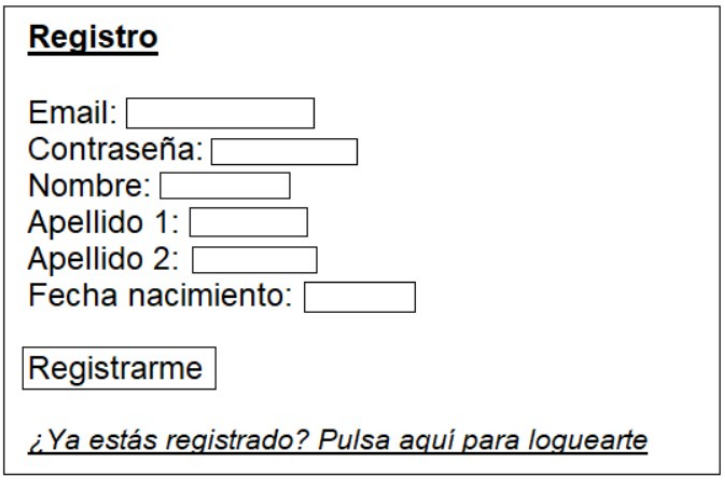
Ejemplo visual de página 1:



Después de introducir un email y contraseña correcta, mostrará los datos…



Ejemplo visual de la página 2:



Práctica 4: Modificación Aplicación de login

Modifique la aplicación “Login” de la Práctica 3 con los siguientes cambios:

- En la página de Registro, añadir un nuevo campo al formulario llamado “Repetir contraseña”. Cuando se pulse el botón de registro, se deberá comprobar que las dos contraseñas introducidas coinciden antes de realizar la inserción en la base de datos. En caso de que las contraseñas no coincidan, se debe mostrar un mensaje de error en rojo “Las contraseñas introducidas no coinciden”
- Crear una nueva página llamada “baja.php”. En ella, se deberá permitir al usuario darse de baja del sistema, borrando cualquier información suya de la base de datos. Como doble verificación, en lugar de pedir solamente email y contraseña para el desregistro, es necesario pedir también el nombre de la persona. Se mostrará el mensaje “El usuario con el email xxxx@xxxx.xx se ha dado de baja correctamente.” en caso de éxito, o el mensaje “La usuario no existe o la contraseña es incorrecta.” en el otro caso. Además, se deberá incluir un vínculo a esta nueva página desde las otras páginas (registro y login) y viceversa.

Práctica 5: HackBlog

En esta práctica se proporciona una maqueta de un mini-blog. El trabajo consiste en añadir el código necesario para que el blog almacene y recupere la información de la base de datos. En los comentarios escritos en los ficheros se indica lo que se debe hacer.

Se deben usar transacciones para insertar comentarios. Utilizar consultas preparadas para consultar la noticia y los comentarios en `noticia.php`.

Se proporcionan los siguientes archivos:

---

bd.php

Este archivo contiene el código que crea la conexión a la BBDD.

```
<?php
// CONECTAR CON LA BBDD USANDO dwes/abc123
// SI HAY ERRORES, SALIR
// USAR UTF-8
```

---

error404.php

Página de error 404.

```
<?php
theHeader("Página no encontrada");
?>
<h2>Error 404</h2>
<h3>Página no encontrada</h3>
<?php
theFooter();
```

---

funciones.php

Funciones varias

```
<?php

function bloqueNoticia($noticia)
{ // REEMPLAZAR TEXTO EN MAYUSCULAS
(ID,TITULAR,ENTRADILLA,FECHA,NUMERO_DE_COMENTARIOS)
 echo "<li class='bloqueNoticia'>";
 echo "<img src='images/thumbnail-ID.jpg' />";
 echo "<h3><a href='noticia.php?id=ID'>TITULAR</a></h3>";
 echo "<p>ENTRADILLA</p>";
 echo "<p><a href='noticia.php?id=ID'>Leer más</a></p>";
 echo "<p>Publicada en: FECHA. (NUMERO_DE_COMENTARIOS) comentarios.</p>";
 echo "</li>";
}

function bloqueComentario($comentario)
{ // REEMPLAZAR TEXTO EN MAYUSCULAS (AUTOR,FECHA,TEXTO)
 echo "<li class='bloqueComentario'>";
 echo "<p><strong>AUTOR</strong> en FECHA:";
 echo "<p>TEXTO</p>";
 echo "</li>";
}
```

```
function theHeader($titulo = null)
{
?>
<!DOCTYPE html>

<html lang="es">
<head>
 <meta charset="utf-8">
 <title><?php if ($titulo) {
 echo $titulo . ' - ';
 }
 ?>The Hack blog</title>
 <link rel="stylesheet" href="style.css" />
</head>

<body>
<header><h1><a href="index.php">The Hack blog</a></h1></header>
<div id="container">
 <div id="main">
<?php
}

function theFooter()
{
?>
 </div>
</div>
<?PHP // REEMPLAZAR TEXTO EN MAYUSCULAS (TU_NOMBRE) ?>
<footer>Copyright © <?php date("Y")?><span>TU_NOMBRE<a href='index.php'>The Hack
Blog</a></span></footer>
</body>
</html>
<?php
}
```

---

hackblog.sql

Contiene el código que crea la base de datos e incluye algunas noticias de ejemplo.

```
-- phpMyAdmin SQL Dump
-- version 4.6.6deb5
-- https://www.phpmyadmin.net/
--
-- Servidor: localhost:3306
-- Tiempo de generación: 22-10-2018 a las 01:07:52
-- Versión del servidor: 5.7.23-0ubuntu0.18.04.1
-- Versión de PHP: 7.2.10-0ubuntu0.18.04.1

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `hackBlog`
--

-- --------------------------------------------------------

--
```

```
-- Estructura de tabla para la tabla `comentarios`
--

CREATE TABLE `comentarios` (
  `id_comentario` int(10) UNSIGNED NOT NULL,
  `autor` varchar(255) COLLATE utf8_bin NOT NULL,
  `texto` text COLLATE utf8_bin NOT NULL,
  `fecha` datetime NOT NULL,
  `id_noticia` int(10) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

--
-- Volcado de datos para la tabla `comentarios`
--

INSERT INTO `comentarios` (`id_comentario`, `autor`, `texto`, `fecha`, `id_noticia`) VALUES
(1, 'echodelta', 'The samba beat comes from two frogs. One big one goes boorup, the other goes be-be-be…..be-be-be. Probably goes on all night long.', '2018-10-01 09:35:33', 3),
(2, 'Gijs', 'i love it.', '2018-10-10 10:38:35', 3);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `noticias`
--

CREATE TABLE `noticias` (
  `id_noticia` int(10) UNSIGNED NOT NULL,
  `titular` varchar(255) COLLATE utf8_bin NOT NULL,
  `entradilla` text COLLATE utf8_bin NOT NULL,
  `cuerpo` text COLLATE utf8_bin NOT NULL,
  `fecha` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

--
-- Volcado de datos para la tabla `noticias`
--

INSERT INTO `noticias` (`id_noticia`, `titular`, `entradilla`, `cuerpo`, `fecha`) VALUES
(1, 'The Crystal (Testing) Method', 'It used to be any good electronics experimenter had a bag full of crystals because you never knew what frequency you might need. These days, you are likely to have far fewer because you usually just need one reference frequency and derive all the other frequencies from it. But how can you test a crystal?', '<p>It used to be any good electronics experimenter had a bag full of crystals because you never knew what frequency you might need. These days, you are likely to have far fewer because you usually just need one reference frequency and derive all the other frequencies from it. But how can you test a crystal? As [Mousa] points out in a recent video, you can’t test it with a multimeter.</p>\r\n\r\n<p>His approach is simple: Monitor a function generator with an oscilloscope, but put the crystal under test in series. Then you move the frequency along until you see the voltage on the oscilloscope peak. That frequency should match the crystal’s operating frequency.</p>\r\n\r\n<p>It is interesting that because of the resonance of the crystal, the voltage on the scope can be much higher than the input voltage from the signal generator. This is a simple test, but effective. Of course, you could also have a little oscillator and see what the crystal does in a real circuit./p>\r\n\r\n<p>We’ve tested crystals before with a network analyzer and we even made a video that shows essentially the same test [Mousa] uses, although it was on an LC circuit, not a crystal. You do need to be careful you aren’t operating the crystal in an overtone mode by accident, although presumably if it works on a harmonic, it should work on the fundamental frequency, too. We’ve also seen crystal testing and classification done with a software defined radio.</p>', '2018-10-20 08:34:26'),
(2, 'I2C Bootloader for ATtiny85 Lets Other Micros Push Firmware Updates', 'There are a few different ways of getting firmware onto one of AVR’s ATtiny85 microcontrollers, including bootloaders that allow for firmware to be updated without the need to plug the chip into a programmer.', '<p>There are a few different ways of getting firmware onto one of AVR’s ATtiny85 microcontrollers, including bootloaders that allow for firmware to be updated without the need to plug the chip into a programmer. However, [casanovg] wasn’t satisfied with those so he sent us a tip letting us know he wrote an I2C bootloader for the ATtiny85 called Timonel. It takes into account a few particulars of the part, such as the fact that it lacks a protected memory area where a bootloader would normally reside, and it doesn’t have a native I2C interface, only the USI (Universal Serial Interface). He’s just released the first functional version for the ATtiny85, but there’s no reason it couldn’t be made to work with the ATtiny45 and ATtiny25 as well.</p>\r\n\r\n<p>Timonel is designed for systems where there is a more powerful microcontroller or microprocessor running the show (such as an ESP8266, Arduino, or even a board like a Raspberry Pi.) In designs where the ATtinys are on an I2C bus performing peripheral functions such as running sensors, Timonel allows the firmware for these
```

```
peripheral MCUs to be updated directly from the I2C bus master. Embedded below is a video demo
of [casanovg] sending simple serial commands, showing a successful firmware update of an AVR
ATtiny85 over I2C.</p>', '2018-10-09 11:20:35'),
(3, 'Hacking Nature’s Musicians', 'We just wrapped up the Musical Instrument Challenge in the
Hackaday Prize, and for most projects that meant replicating sounds made by humans, or
otherwise making musicians for humans. There’s more to music than just what can be made in a
DAW, though.', '<p>We just wrapped up the Musical Instrument Challenge in the Hackaday Prize,
and for most projects that meant replicating sounds made by humans, or otherwise making
musicians for humans. There’s more to music than just what can be made in a DAW, though; the
world is surrounded by a soundscape, and you only need to take a walk in the country to hear
it.</p>\r\n\r\n<p>For her Hackaday Prize entry, [Kelly] is hacking nature’s musicians. She’s
replicating the sounds of the rural countryside in transistors and PCBs. It’s an astonishing
work of analog electronics, and it sounds awesome, too.</p>\r\n\r\n<p>The most impressive
board [Kelly] has been working on is the Mother Nature Board, a sort of natural electronic
chorus of different animal circuits. It’s all completely random, based on a Really, Really
Random Number Generator, and uses a collection of transistors and 555 timers to create pulses
sent to a piezo. This circuit is very much sensitive to noise, and while building it [Kelly]
found that not all of her 2N3904 transistors were the same; some of them worked for the noise
generator, some didn’t. This is a tricky circuit to design, but the results are
delightful.</p>\r\n\r\n<p>So, can analog electronics sound like a forest full of crickets?
Surprisingly, yes. This demonstration shows what’s possible with a few breadboards full of
transistors, caps, resistors, and LEDs. It’s an electronic sculpture of the sounds inspired by
the nocturnal soundscape of rural Virginia. You’ve got crickets, cicadas, katydids, frogs,
birds, and all the other non-human musicians in the world. Beautiful.</p>', '2018-09-19
18:30:46');

--
-- Índices para tablas volcadas
--

--
-- Indices de la tabla `comentarios`
--
ALTER TABLE `comentarios`
  ADD PRIMARY KEY (`id_comentario`),
  ADD KEY `FK_noticias` (`id_noticia`);

--
-- Indices de la tabla `noticias`
--
ALTER TABLE `noticias`
  ADD PRIMARY KEY (`id_noticia`);

--
-- AUTO_INCREMENT de las tablas volcadas
--

--
-- AUTO_INCREMENT de la tabla `comentarios`
--
ALTER TABLE `comentarios`
  MODIFY `id_comentario` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
--
-- AUTO_INCREMENT de la tabla `noticias`
--
ALTER TABLE `noticias`
  MODIFY `id_noticia` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
--
-- Restricciones para tablas volcadas
--

--
-- Filtros para la tabla `comentarios`
--
ALTER TABLE `comentarios`
  ADD CONSTRAINT `FK_noticias` FOREIGN KEY (`id_noticia`) REFERENCES `noticias` (`id_noticia`)
ON DELETE CASCADE ON UPDATE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

index.php

Página de portada.

```
<?php
require_once "funciones.php";
require_once "bd.php";

theHeader();
?>

<h2>Últimas entradas</h2>
<ul class="listaNoticias">
<?php
/* OBTENER DE LA BD LA LISTA DE NOTICIAS Y PASARSELAS A bloqueNoticia. OBTENER
TAMBIÉN TOTAL DE COMENTARIOS */
for ($i = 0; $i < 3; $i++) { // BUCLE DE MUESTRA
    $noticia = "";
    bloqueNoticia($noticia);
}
?>
</ul>
<?php
theFooter();
```

---

noticia.php

Página de noticia.

```
<?php
require_once "funciones.php";
require_once "bd.php";

if (isset($_POST['btnComentario'])) {
    // INSERTAR COMENTARIO
}
// CONSULTAR NOTICIA
if (false) { /* COMPROBAR SI NO HA HABIDO RESULTADOS. SI ES ASÍ, LANZAMOS ERROR 404*/
    header($_SERVER["SERVER_PROTOCOL"] . " 404 Not Found", true, 404);
    include 'error404.php';
    exit();
}
// OBTENER DATOS NOTICIA
// REEMPLAZAR TEXTO EN MAYUSCULAS (TITULAR,FECHA,ID,CUERPO)
theHeader("TITULAR");
?>
    <p class="small"><a href="index.php">← Volver a portada</a></p>
    <h2>TITULAR</h2>
    <p class="small">Publicado en: FECHA</p>
    <div id="imgcover"><img src='images/cover-ID.jpg' /></div>
    CUERPO


    <h2>Comentarios</h2>
    <?php
// OBTENER COMENTARIOS DE LA NOTICIA
if (true) { // SI HAY COMENTARIOS
    echo '<ul>';
    for ($i = 0; $i < 3; $i++) { // BUCLE DE MUESTRA
        $comentario = "";
        bloqueComentario($comentario);
    }
    echo '</ul>';
} else {
    echo "<p>Sin comentarios</p>";
```

```
}
?>
    <strong>Deja un comentario</strong>
    <form id="frmComentario" method="post" action="<?php echo $_SERVER['PHP_SELF'] .
'?' . $_SERVER['QUERY_STRING'] ?>">
    <input name="hidNoticia" type="hidden" value="<?php echo $_GET['id']; ?>" />
    <p><input name="txtAutor" type="text" placeholder="Nombre" /></p>
    <p><textarea placeholder="Escribe tu comentario"
name="txtComentario"></textarea></p>
    <p><input type="submit" name="btnComentario" value="Comentar" /></p>
    </form>
    <?php
theFooter();
```

---

style.css

Hoja de estilos

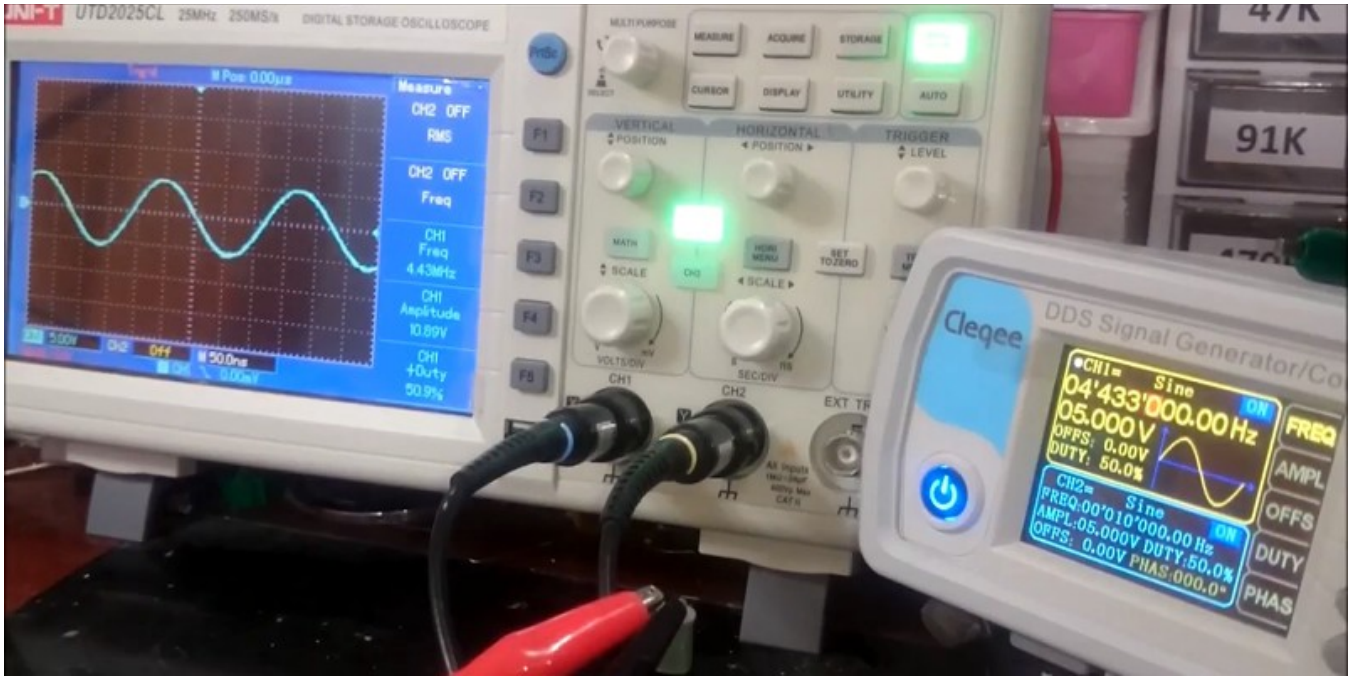
```
#main {
    width: 1000px;
    margin: auto;
    border-left: 1px solid #666;
    border-right: 1px solid #666;
    background-color: #ffffff;
    padding: 4px 20px;
}
body {
    background-color: #333;
    margin: 0px;
    font-family: Verdana,sans-serif;
}
#container {
    background-color: #EEE;

}

header {
    background-color: #333;
    color: #EEE;
    padding: 2px 10px;
    border-bottom: 5px solid #ad8b0f;
    text-align: center;
}
header h1 {
    margin: 0px;
    font-size: 40px;
}
h1, h2, h3, h4 {
    font-family: "Segoe UI",Arial,sans-serif;
    text-transform: uppercase;
}

a {
    color: #ad8b0f;
    text-decoration: none;
}
header a, footer a {
    color: #EEE;
}
h2>a, h3>a {
    font-weight: 800;
    color: #333;
}

#main h2 {
    font-size: 30px;
}

#h3 {
    font-size: 20px;
```

```
}

.listaNoticias {
    padding:0px 20px;
}

.bloqueNoticia img {
    float: left;
    border: 1px solid #666;
    margin-right: 10px;
    width: 200px;
    height: 200px;
}

.bloqueNoticia {
    clear: both;
    list-style: none;
    margin-bottom: 10px;
    min-height: 200px;
    border-left: 3px dashed #ad8b0f;
    padding-left: 10px;
}

#imgcover {
        text-align: center;
        background-color: #EEE;
}

.small {
    font-size: 10px;
}

footer {
    border-top: 5px solid #ad8b0f;
    text-align: center;
    padding: 5px 5px;
    color: #EEE;
    font-size: 10px;
}

footer span {
    padding-left: 5px;
    border-left: 1px solid #ad8b0f;
}


.bloqueComentario {
    list-style: none;
    margin-bottom: 10px;
    border-left: 3px dashed #ad8b0f;
    padding-left: 10px;
}

#frmComentario textarea {
    width: 100%;
}
```

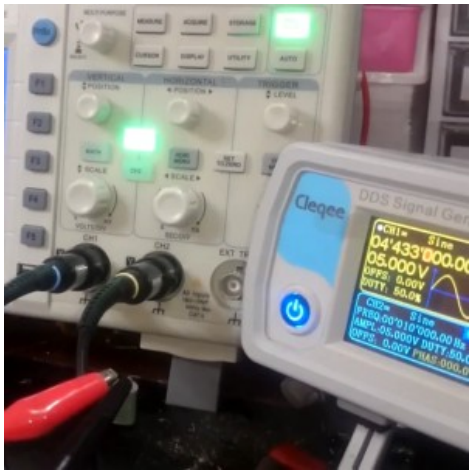
Carpeta images

Carpeta que contiene las imágenes destacadas de las noticias (cover) y sus miniaturas (thumbnail):

cover-1.jpg

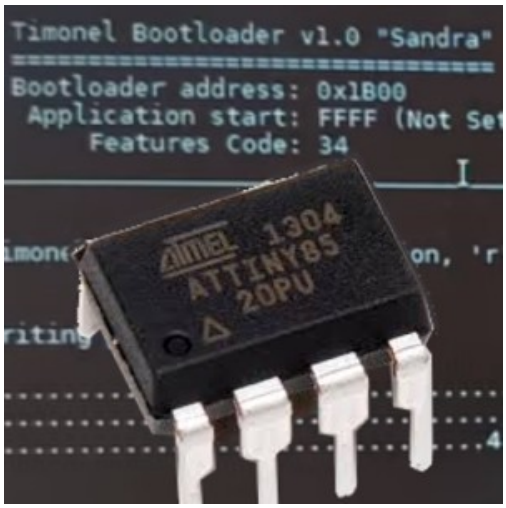


thumbnail-1.jpg



cover-2.jpg

thumbnail-2.jpg



cover-3.jpg

thumbnail-3.jpg

Ejercicios de refuerzo

Refuerzo 1: IMDWes

La página imita la ficha de una película en una web de cine, en la que se muestran los datos de la película y un listado de opiniones extraídas de la base de datos.

Se pide lo siguiente:

- Crear una conexión con la base de datos utilizando mysqli o PDO.
- Obtener las opiniones de la base de datos y mostrarlas en el listado, rellenándolo según el código de ejemplo. Si no hay opiniones, se mostrará el texto "Sin opiniones".
- Completar el formulario para que el usuario pueda enviar su opinión. Se debe procesar en la misma página y debe generar los elementos del `SELECT` con un bucle.
- Al procesar el envío del formulario, si el usuario está permitido la opinión debe introducirse en la base de datos usando transacciones. De lo contrario debe mostrar un mensaje de error.
- La inserción de datos deberá hacerse usando transacciones
- Creación de función `filtro()`: recibe el nombre del usuario que hace el comentario y comprueba si está en un array con nombres de usuarios prohibidos que la función contiene. Devuelve true si el usuario está en la lista negra y falso en caso contrario.
- Creación de función `claseNota()`: se utiliza en la lista de opiniones para escribir la clase que le corresponde a la nota. Recibe una nota numérica y devuelve una cadena con la clase que le corresponde a dicha nota según la siguiente regla:
    - 0, 1, 2 o 3: "peli_mala"
    - 4, 5 o 6: "peli_regular"
    - 7 u 8: "peli_buena"
    - 9 o 10: "peli_muy_buena".
- Las funciones se crearán en un fichero aparte llamado funciones.php que deberá ser importado.
- Utiliza comentarios en las funciones desarrolladas.

Se aportan los siguientes archivos como estructura base

---

estilo.css

```
body {
    position: static;
    width: auto;
    margin: auto;
    min-width: 1008px;
    background: #b3b3b0;
}

#wrapper {
    font-family: Verdana, Arial, sans-serif;
    color:#333;
    font-size: 13px;
    width: 1008px;
    margin: auto;
    background-color: #fff;
    box-shadow: 0px 0px 8px rgba(0,0,0,0.7);
    box-sizing: border-box;
}

.header {
    background: #191919;
    border-top: 1px solid #444;
    color: #fff;
    font: 10pt Verdana, Arial, sans-serif;
    padding: 15px;
}
```

```
.logo {
    background-color: #F5C518;
    border-color: #F5C518;
    padding: 5px 10px 5px 10px;
    display: inline-block;
    margin-left: 0px;
    font: 36px Arial,sans-serif;
    color:#000;
    font-weight: bold;
    border-radius: 4px;
}

.opiniones {
    list-style: none;
}

.opinion {
    margin-bottom: 5px;
}

.opinion .nombre {
    text-transform: uppercase;
    font-weight: bold;
}

.opinion .nota {
    display: inline-block;
    border-radius: 5px;
    padding: 1px;
    color: black;
    font-weight: bold;
}

.opinion .fecha {
    font-style: italic;
    color: #333;
    font-size: smaller;
}

.nota.peli_mala {
    background-color: red;
}

.nota.peli_regular {
    background-color: yellow;
}

.nota.peli_buena {
    background-color: green;
}

.nota.peli_muy_buena {
    background-color: greenyellow;
}

.superior {
    background-color:#333;
    padding: 5px 15px;
    color: #fff;
}

.superior h1 {
    margin-top: 0px;
}

.poster {
    margin: auto;
    display: block;
    box-shadow: 0px 0px 8px rgba(0,0,0,0.7);
}

.inferior {
    padding: 15px;
}
```

funciones.php

```
<?php

// Dentro de la función que comprueba si el nombre de usuario está permitido se ha de usar la siguiente lista negra:
$listaNegra = array("Manolo29", "LuciaUchiha", "Ecijano42", "usuario", "root");
```

IMDWes.sql

```
-- phpMyAdmin SQL Dump
-- version 4.6.6deb5
-- https://www.phpmyadmin.net/
--
-- Servidor: localhost:3306
-- Tiempo de generación: 22-10-2018 a las 01:07:52
-- Versión del servidor: 5.7.23-0ubuntu0.18.04.1
-- Versión de PHP: 7.2.10-0ubuntu0.18.04.1

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `IMDwes`
--

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `comentarios`
--

CREATE TABLE `opiniones` (
  `id_opinion` int(10) UNSIGNED NOT NULL,
  `usuario` varchar(255) COLLATE utf8_bin NOT NULL,
  `texto` text COLLATE utf8_bin NOT NULL,
  `fecha` datetime NOT NULL,
  `nota` int(10) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

--
-- Volcado de datos para la tabla `comentarios`
--

INSERT INTO `opiniones` (`id_opinion`, `usuario`, `texto`, `fecha`, `nota`) VALUES
(1, 'Susana23', '¡¡Me ha encantado!!', '2018-10-01 09:35:33', 8);

-- --------------------------------------------------------

--
-- Índices para tablas volcadas
--

--
-- Indices de la tabla `comentarios`
--
ALTER TABLE `opiniones`
  ADD PRIMARY KEY (`id_opinion`);

--
-- AUTO_INCREMENT de las tablas volcadas
--

--
-- AUTO_INCREMENT de la tabla `comentarios`
--
ALTER TABLE `opiniones`
  MODIFY `id_opinion` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
```

```
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

---

index.php

```
<html>
    <head>
        <title>Piratas de Silicon Valley (1999) - IMDwes</title>
        <link rel="stylesheet" href="estilo.css">
    </head>
<body>
    <div id="wrapper">
        <div class="header"><span class="logo">IMDwes</span></div>
        <div class="superior">
            <h1>Piratas de Silicon Valley (1999)</h1>
            <p>1h 35min | Biografía, Drama, Historia | 20 Junio 1999</p>
        </div>
        <div class="inferior">
            <img class="poster" src="img/poster.jpg" />
            <p><strong>Sinopsis</strong>: Los logros de los visionarios Steve Jobs y
Bill Gates revolucionan el siglo XX, e inician el camino que lleva al estado actual
del mundo de la computación y la informática.</p>
            <h2>Opiniones</h2>
            <ul class="opiniones">
                <li class="opinion">
                    <span class="nota CLASENOTA">NOTA</span>
                    <span class="nombre">NOMBRE</span>
                    OPINIÓN
                    <span class="fecha">FECHA</span>
                </li>
                <li class="opinion">
                    <span class="nota CLASENOTA">NOTA</span>
                    <span class="nombre">NOMBRE</span>
                    OPINIÓN
                    <span class="fecha">FECHA</span>
                </li>
                <li class="opinion">
                    <span class="nota CLASENOTA">NOTA</span>
                    <span class="nombre">NOMBRE</span>
                    OPINIÓN
                    <span class="fecha">FECHA</span>
                </li>
            </ul>
            <h3>Añade tu opinión</h3>
            <form>
                <p><input type="text" placeholder="Nombre" /></p>
                <p>Nota: <select></select></p>
                <p><textarea placeholder="Escribe tu opinión"></textarea></p>
                <p><input type="submit" value="Enviar" /></p>
            </form>
            </div>
        </div>
    </body>
</html>
```

Ejercicios de ampliación

Ampliación 1: Extensión mysqli

(Práctica con todo el código de estas explicaciones realizando varias pruebas de ejecución)

Esta extensión se desarrolló para aprovechar las ventajas que ofrecen las versiones 4.1.3 y posteriores de MySQL, y viene incluida con PHP a partir de la versión 5. Ofrece un interface de programación dual, pudiendo accederse a las funcionalidades de la extensión utilizando objetos o funciones de forma indiferente. Por ejemplo, para establecer una conexión con un servidor MySQL y consultar su versión, podemos utilizar cualquiera de las siguientes formas:

```
// utilizando constructores y métodos de la programación orientada a objetos
$conexion = new mysqli('localhost', 'usuario', 'contraseña', 'base_de_datos');
print $conexion->server_info;

// utilizando llamadas a funciones
$conexion = mysqli_connect('localhost', 'usuario', 'contraseña', 'base_de_datos');
print mysqli_get_server_info($conexion);
```

En ambos casos, la variable `$conexion` es de tipo objeto. La utilización de los métodos y propiedades que aporta la clase mysqli normalmente produce un código más corto y legible que si utilizas llamadas a funciones.

Para saber más ...

Toda la información relativa a la instalación y utilización de la extensión, incluyendo las funciones y métodos propios de la extensión, se puede consultar en el manual de PHP.

Entre las mejoras que aporta a la antigua extensión mysql, figuran:

- Interface orientado a objetos.
- Soporte para transacciones.
- Soporte para consultas preparadas.
- Mejores opciones de depuración.

Como ya viste en la primera unidad, las opciones de configuración de PHP se almacenan en el fichero php.ini. En este fichero hay una sección específica para las opciones de configuración propias de cada extensión. Entre las opciones que puedes configurar para la extensión MySQLi están:

- mysqli.allow_persistent. Permite crear conexiones persistentes.
- mysqli.default_port. Número de puerto TCP predeterminado a utilizar cuando se conecta al servidor de base de datos.
- mysqli.reconnect. Indica si se debe volver a conectar automáticamente en caso de que se pierda la conexión.
- mysqli.default_host. Host predeterminado a usar cuando se conecta al servidor de base de datos.
- mysqli.default_user. Nombre de usuario predeterminado a usar cuando se conecta al servidor de base de datos.
- mysqli.default_pw. Contraseña predeterminada a usar cuando se conecta al servidor de base de datos.

Para saber más ...

En la documentación de PHP se incluye una lista completa de las directivas relacionadas con la extensión MySQLi que se pueden utilizar en php.ini.

Establecimiento de conexiones

Para poder comunicarte desde un programa PHP con un servidor MySQL, el primer paso es establecer una conexión. Toda comunicación posterior que tenga lugar, se hará utilizando esa conexión.

Si utilizas la extensión MySQLi, establecer una conexión con el servidor significa crear una instancia de la clase mysqli. El constructor de la clase puede recibir seis parámetros, todos opcionales, aunque lo más habitual es utilizar los cuatro primeros:

- El nombre o dirección IP del servidor MySQL al que te quieres conectar.
- Un nombre de usuario con permisos para establecer la conexión.
- La contraseña del usuario.
- El nombre de la base de datos a la que conectarse.
- El número del puerto en que se ejecuta el servidor MySQL.
- El socket o la tubería con nombre (named pipe) a usar.

Si utilizas el constructor de la clase, para conectarte a la base de datos "dwes" puedes hacer:

```
// utilizando el constructor de la clase
$bd = new mysqli('localhost', 'dwes', 'abc123', 'dwes');
```

Aunque también tienes la opción de primero crear la instancia, y después utilizar el método connect para establecer la conexión con el servidor:

```
// utilizando el método connect
$bd = new mysqli();
$bd->connect('localhost', 'dwes', 'abc123', 'dwes');
```

Por el contrario, utilizando el interface procedimental de la extensión:

```
// utilizando llamadas a funciones
$bd = mysqli_connect('localhost', 'dwes', 'abc123', 'dwes');
```

Es importante verificar que la conexión se ha establecido correctamente. Para comprobar el error, en caso de que se produzca, puedes usar las siguientes propiedades (o funciones equivalentes) de la clase mysqli:

- connect_errno (o la función mysqli_connect_errno) devuelve el número de error o null si no se produce ningún error.
- connect_error (o la función mysqli_connect_error) devuelve el mensaje de error o null si no se produce ningún error.

Por ejemplo, el siguiente código comprueba el establecimiento de una conexión con la base de datos "dwes" y finaliza la ejecución si se produce algún error:

```
@$bd = new mysqli('localhost', 'dwes', 'abc123', 'dwes');
$error = $bd->connect_errno;
if ($error != null) {
     echo "<p>Error $error conectando a la base de datos: $bd->connect_error</p>";
     exit();
}
```

En PHP, como veremos posteriormente con más detalle, puedes anteponer a cualquier expresión el operador de control de errores @ para que se ignore cualquier posible error que pueda producirse al ejecutarla.

Si una vez establecida la conexión, quieres cambiar la base de datos puedes usar el método select_db (o la función mysqli_select_db de forma equivalente) para indicar el nombre de la nueva.

```
// utilizando el método connect
$bd->select_db('otra_bd');
```

Una vez finalizadas las tareas con la base de datos, utiliza el método close (o la función mysqli_close) para cerrar la conexión con la base de datos y liberar los recursos que utiliza.

```
$bd->close();
```

Actividad

Crea un script que se conecte a la base de datos y compruebe si se ha conectado sin errores. Cierra la conexión al terminar.

Ejecución de consultas

La forma más inmediata de ejecutar una consulta, si utilizas esta extensión, es el método query, equivalente a la función mysqli_query. Si se ejecuta una consulta de acción que no devuelve datos (como una sentencia SQL de tipo UPDATE, INSERT o DELETE), la llamada devuelve true si se ejecuta correctamente o false en caso contrario. El número de registros afectados se puede obtener con la propiedad affected_rows (o con la función mysqli_affected_rows).

```
@$bd = new mysqli('localhost', 'dwes', 'abc123', 'dwes');
$error = $bd->connect_errno;
if ($error == null) {
  $resultado = $bd->query('DELETE FROM stock WHERE unidades=0');
  if ($resultado) {
    echo "<p>Se han borrado $bd->affected_rows registros.</p>";
  }
  $bd->close();
}
```

En el caso de ejecutar una sentencia SQL que sí devuelva datos (como un SELECT), éstos se devuelven en forma de un objeto resultado (de la clase mysqli_result). En el punto siguiente verás cómo se pueden manejar los resultados obtenidos.

El método query tiene un parámetro opcional que afecta a cómo se obtienen internamente los resultados, pero no a la forma de utilizarlos posteriormente. En la opción por defecto, MYSQLI_STORE_RESULT, los resultados se recuperan todos juntos de la base de datos y se almacenan de forma local. Si cambiamos esta opción por el valor MYSQLI_USE_RESULT, los datos se van recuperando del servidor según se vayan necesitando.

```
$resultado = $bd->query('SELECT producto, unidades FROM stock', MYSQLI_USE_RESULT);
```

Para saber más ...

Otra forma que puedes utilizar para ejecutar una consulta es el método real_query (o la función mysqli_real_query), que siempre devuelve true o false según se haya ejecutado correctamente o no. Si la consulta devuelve un conjunto de resultados, se podrán recuperar de forma completa utilizando el método store_result, o según vaya siendo necesario gracias al método use_result.

Es importante tener en cuenta que los resultados obtenidos se almacenarán en memoria mientras los estés usando. Cuando ya no los necesites, los puedes liberar con el método free de la clase mysqli_result (o con la función mysqli_free_result):

```
$resultado→free();
```

Transacciones

Como ya comentamos, si necesitas utilizar transacciones deberás asegurarte de que estén soportadas por el motor de almacenamiento que gestiona tus tablas en MySQL. Si utilizas InnoDB, por defecto cada consulta individual se incluye dentro de su propia transacción. Puedes gestionar este comportamiento con el método autocommit (función mysqli_autocommit).

Al deshabilitar las transacciones automáticas, las siguientes operaciones sobre la base de datos iniciarán una transacción que deberás finalizar utilizando:

- **commit** (o la función `mysqli_commit`). Realizar una operación "commit" de la transacción actual, devolviendo true si se ha realizado correctamente o false en caso contrario.
- **rollback** (o la función `mysqli_rollback`). Realizar una operación "rollback" de la transacción actual, devolviendo true si se ha realizado correctamente o false en caso contrario.

```
$bd->autocommit(false);
if ($bd->query('DELETE FROM stock WHERE unidades=0') == true) {
  if ($bd->query('UPDATE stock SET unidades=3 WHERE producto="STYLUSSX515W"')) {
    $bd->commit();  // Confirma los cambios
  } else {
      $bd->rollback(); // Revierte los cambios
  }
} else {
    $bd->rollback();  // Revierte los cambios
}
```

Una vez finalizada esa transacción, comenzará otra de forma automática.

Resultados

Ya sabes que al ejecutar una consulta que devuelve datos obtienes un objeto de la clase mysqli_result. Esta clase sigue los criterios de ofrecer un interface de programación dual, es decir, una función por cada método con la misma funcionalidad que éste.

Para trabajar con los datos obtenidos del servidor, tienes varias posibilidades:

- **fetch_array** (función `mysqli_fetch_array`). Obtiene un registro completo del conjunto de resultados y lo almacena en un array. Por defecto el array contiene tanto claves numéricas como asociativas. Por ejemplo, para acceder al primer campo devuelto, podemos utilizar como clave el número 0 o su nombre indistintamente. Este comportamiento por defecto se puede modificar utilizando un parámetro opcional, que puede tomar los siguientes valores:
  - **MYSQLI_NUM**. Devuelve un array con claves numéricas.
  - **MYSQLI_ASSOC**. Devuelve un array asociativo.
  - **MYSQLI_BOTH**. Es el comportamiento por defecto, en el que devuelve un array con claves numéricas y asociativas.

```
$resultado = $bd->query('SELECT producto, unidades FROM stock WHERE
unidades<2');
$stock = $resultado->fetch_array();  // Obtenemos el primer registro
```

```
$producto = $stock['producto'];  // O también $stock[0];
$unidades = $stock['unidades'];  // O también $stock[1];
print "<p>Producto $producto: $unidades unidades.</p>";
```

- **fetch_assoc** (función `mysqli_fetch_assoc`). Idéntico a fetch_array pasando como parámetro `MYSQLI_ASSOC`.
- **fetch_row** (función `mysqli_fetch_row`). Idéntico a fetch_array pasando como parámetro `MYSQLI_NUM`.
- **fetch_object** (función `mysqli_fetch_object`). Similar a los métodos anteriores, pero devuelve un objeto en lugar de un array. Las propiedades del objeto devuelto se corresponden con cada uno de los campos del registro.

Parar recorrer todos los registros de un array, puedes hacer un bucle teniendo en cuenta que cualquiera de los métodos o funciones anteriores devolverá null cuando no haya más registros en el conjunto de resultados.

```
$resultado = $bd->query('SELECT producto, unidades FROM stock WHERE unidades<2');
$stock = $resultado->fetch_object();
while ($stock != null) {
    print "<p>Producto $stock->producto: $stock->unidades unidades.</p>";
    $stock = $resultado->fetch_object();
}
```

Otra forma de hacerlo:

```
$resultado = $bd->query('SELECT producto, unidades FROM stock WHERE unidades<2');
while ($stock = $resultado->fetch_object()) {
    print "<p>Producto $stock->producto: $stock->unidades unidades.</p>";
}
```

Para saber más

En el manual de PHP tienes más información sobre los métodos y propiedades de la clase mysqli_result.

Consultas preparadas

Cada vez que se envía una consulta al servidor, éste debe analizarla antes de ejecutarla. Algunas sentencias SQL, como las que insertan valores en una tabla, deben repetirse de forma habitual en un programa. Para acelerar este proceso, MySQL admite consultas preparadas. Estas consultas se almacenan en el servidor listas para ser ejecutadas cuando sea necesario.

Se mejora la eficiencia, ya que el análisis de la sentencia se lleva a cabo sólo una vez. Además, reciben los parámetros por referencia, por lo que también son más resistentes a la inyección SQL.

Para trabajar con consultas preparadas con la extensión MySQLi de PHP, debes utilizar la clase mysqli_stmt. Utilizando el método stmt_init de la clase mysqli (o la función mysqli_stmt_init) obtienes un objeto de dicha clase.

```
$bd = new mysqli('localhost', 'dwes', 'abc123', 'dwes');
$consulta = $bd->stmt_init();
```

Los pasos que debes seguir para ejecutar una consulta preparada son:

- Preparar la consulta en el servidor MySQL utilizando el método prepare (función `mysqli_stmt_prepare`).

- Ejecutar la consulta, tantas veces como sea necesario, con el método execute (función `mysqli_stmt_execute`).
- Una vez que ya no se necesita más, se debe ejecutar el método close (función `mysqli_stmt_close`).

Por ejemplo, para preparar y ejecutar una consulta que inserta un nuevo registro en la tabla familia:

```
$consulta = $bd->stmt_init();
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES ("TABLET", "Tablet
PC")');
$consulta->execute();
$consulta->close();
$bd->close();
```

El problema que ya habrás observado, es que de poco sirve preparar una consulta de inserción de datos como la anterior, si los valores que inserta son siempre los mismos. Por este motivo las consultas preparadas admiten parámetros. Para preparar una consulta con parámetros, en lugar de poner los valores debes indicar con un signo de interrogación su posición dentro de la sentencia SQL.

```
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
```

Y antes de ejecutar la consulta tienes que utilizar el método **bind_param** (o la función `mysqli_stmt_bind_param`) para sustituir cada parámetro por su valor. El primer parámetro del método `bind_param` es una cadena de texto en la que cada carácter indica el tipo de un parámetro, según la siguiente tabla:

Carácter	Tipo del parámetro
I	Número entero
D	Número real (doble precisión)
S	Cadena de texto
B	Contenido en formato binario (BLOB)

En el caso anterior, si almacenas los valores a insertar en sendas variables, puedes hacer:

```
$consulta = $bd->stmt_init();
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
$cod_producto = "TABLET";
$nombre_producto = "Tablet PC";
$consulta->bind_param('ss', $cod_producto, $nombre_producto);
$consulta->execute();
$consulta->close();
$bd->close();
```

Cuando uses `bind_param` para enlazar los parámetros de una consulta preparada con sus respectivos valores, deberás usar siempre variables como en el ejemplo anterior. Si intentas utilizar literales, por ejemplo:

```
$consulta->bind_param('ss', 'TABLET', 'Tablet PC');  // Genera un error
```

Obtendrás un error. El motivo es que los parámetros del método `bind_param` se pasan por referencia. Aprenderás a usar paso de parámetros por referencia en una unidad posterior.

El método `bind_param` permite tener una consulta preparada en el servidor MySQL y ejecutarla tantas veces como quieras cambiando ciertos valores cada vez. Además, en el caso de las consultas que devuelven valores, se puede utilizar el método **bind_result** (función `mysqli_stmt_bind_result`)

para asignar a variables los campos que se obtienen tras la ejecución. Utilizando el método fetch (`mysqli_stmt_fetch`) se recorren los registros devueltos. Observa el siguiente código:

```
$consulta = $bd->stmt_init();
$consulta->prepare('SELECT producto, unidades FROM stock WHERE unidades<2');
$consulta->execute();
$consulta->bind_result($producto, $unidades);
while($consulta->fetch()) {
    print "<p>Producto $producto: $unidades unidades.</p>";
}
$consulta->close();
$bd->close();
```

Ampliación 2: Encriptar contraseñas de Práctica 3

Modifica la Aplicación Login (Práctica 3) para que las contraseñas estén encriptadas en la base de datos.

Ejemplo: imagina que quieres guardar la contraseña “hola123” en la base de datos. En lugar de almacenar esa cadena tal cual, se almacenará su versión encriptada.

¿Cómo se puede encriptar una cadena en PHP?

```
$encriptada = password_hash(“hola123”, PASSWORD_DEFAULT);
```

Lo que guardaremos en la base de datos es la cadena $encriptada.

¿Cómo comprobamos que una contraseña sin encriptar coincide con una encriptada?

Suponiendo que el usuario a través del formulario ha introducido una contraseña $contraseñaSinEncriptar, y que en la base de datos tenemos una $contraseñaEncriptada:

```
$correcta = password_verify($contraseñaSinEncriptar, $contraseñaEncriptada;
```

$correcta es un booleano que indica si las contraseñas coinciden.

Ampliación 3: Añadir noticia

Añade a la práctica HackBlog una página `anadir_noticia.php` en la que, con un formulario, puedas añadir una noticia nueva.

El formulario debe permitir también adjuntar imágenes (se supone que las envían en formato jpg).

Se proponen dos opciones:

- Se permitirá subir dos imágenes, una para la noticia (cover) y otra para la portada (thumbnail).
- Se permitirá subir una imagen y, a partir de esta se crearán las copias redimensionadas con el tamaño adecuado para la noticia (cover) y para la portada (thumbnail).