UNIVERSIDAD CATÓLICA DE SANTA MARÍA
PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS

# SESIÓN N° 08:

# PHP y MySQL

## I
## OBJETIVOS

- Conocer las características de PHP y My-SQL.
- Desarrollar aplicaciones con PHP
- Conocer las características de MySQL
- Desarrollar aplicaciones con MySQL

## II
## TEMAS A TRATAR

- Introducción.
- PHP
- MySQL
- Resumen

## III
## MARCO TEORICO

## 1. INTRODUCCIÓN

PHP es el lenguaje de lado servidor más extendido en la web. Nacido en 1994, se trata de un lenguaje de creación relativamente reciente. Es un lenguaje que ha tenido una gran aceptación en la comunidad de desarrolladores, debido a la potencia y simplicidad que lo caracterizan, así como al soporte generalizado en la mayoría de los servidores de hosting, hasta los más simples y económicos.

La facilidad de PHP se basa en que permite embeber pequeños fragmentos de código dentro de lo que sería una página común creada con HTML. Esos scripts PHP nos permiten realizar determinadas acciones de una forma fácil y eficaz, pudiendo realizar todo tipo de tareas, de las más simples a las más complejas. Esta combinación de PHP dentro del marco de un documento HTML es lo que permite a desarrolladores sin prácticamente nada de experiencia crear comportamientos atractivos de una manera sencilla, una de las claves del éxito del lenguaje. En resumen, con PHP escribimos scripts dentro del código HTML. Como ya estamos familiarizados con HTML, empezar a desarrollar con PHP es prácticamente inmediato. Por otra parte, y es aquí donde reside su mayor interés, PHP ofrece un sinfín de funciones para la explotación de todo tipo de recursos, entre los que destacan las bases de datos, a las que podremos acceder de una manera llana, sin complicaciones.

PHP es lo que se denomina una tecnología del lado del servidor, que ahora se suele englobar dentro del término "Backend". Existen diversos competidores de PHP en el mundo Backend y todos tienen sus cosas buenas y malas. Resultaría muy arriesgado decir que una tecnología o un lenguaje sea mejor o peor que otro, pero sí podemos decir que PHP es el lenguaje preferido por el mayor número de programadores dedicados en el área Backend. Como competidores de PHP podríamos mencionar ASP.NET (o ASP tradicional), NodeJS, Ruby, Java, Python y un largo etc. Sin embargo, en nuestra opinión, si lo que quieres es desarrollar páginas web, el más sencillo y directo con el que podrías empezar es PHP.

Otra de las claves del éxito de PHP es que la mayoría de los CMS más populares (WordPress,

Joomla!, Drupal) y los sistemas de comercio electrónico (Prestashop, Woocommerce, Magento), así como otros cientos de herramientas, están desarrollados en PHP. Por lo tanto, usar PHP es sinónimo de ser capaz de introducirte en muchas herramientas gratuitas y de código abierto para realizar cualquier cosa en el ámbito de la web.

Mysql es un sistema de base de datos de uso muy extendido a través de internet junto con Apache, este tipo de base de datos es lo que se llama habitualmente Structured Query Language (SQL) siendo un tipo de base de datos lo suficientemente flexible y robusto que a su vez nos permite introducir y consultar grandes cantidades de datos de manera muy simple, una ventaja son las distintas formas de realizar las conexiones y acciones sobre el servidor ya sea desde ODBC hasta PHP (pasando por un montón de lenguajes tanto web como de clientes).

Otra ventaja es que es gratuito en las plataformas Unix (ya sea algún clon gratuito de BSD o Linux dado que viene dentro de la distribución) pero deja de serlo cuando vendemos alguna aplicación con base en el o lo utilizamos para fines comerciales pero la ventaja de poder funcionar sobre un S.O libre y gratuito ya que la licencia es muy baja con respecto a otras bases de datos permitiendo tener un muy bajo costo contra el costo que pueden ofrecer otras compañías. Entre sus virtudes también tenemos una gran performance, el tema de los costos dado que corre en cualquier pc o servidor (como mencionamos anteriormente), ofrece conectividad y seguridad, gran capacidad de procesamiento, portabilidad y distribución abierta.

Mysql nos proporciona algunas herramientas básicas, entre ellas: el servidor, un cliente y distintas API para una amplia variedad de lenguajes, el más utilizado es el de PHP con Apache, donde existe una instalación llamada LAMP, que instala Apache, Mysql y PHP en la mayoría de las distribuciones de los Linux el cual pueden ver como instalarlo en este post.

## 2. PHP

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

Para agregar un programa PHP dentro de una página HTML debemos por un lado al crear el archivo definirlo con extensión php (a diferencia de las páginas estáticas que tienen extensión htm o html) y dentro del contenido de la página, encerrar el programa entre los símbolos <?php [aqui el programa PHP] ?>.

WampServer (Apache-MySQL-PHP): Este software contiene todo lo que necesitamos para probar en forma local en nuestro equipo los programas que codifiquemos en PHP.

## 3. MYSQL

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.

Está desarrollado en su mayor parte en ANSI C y C++. Tradicionalmente se considera uno de los

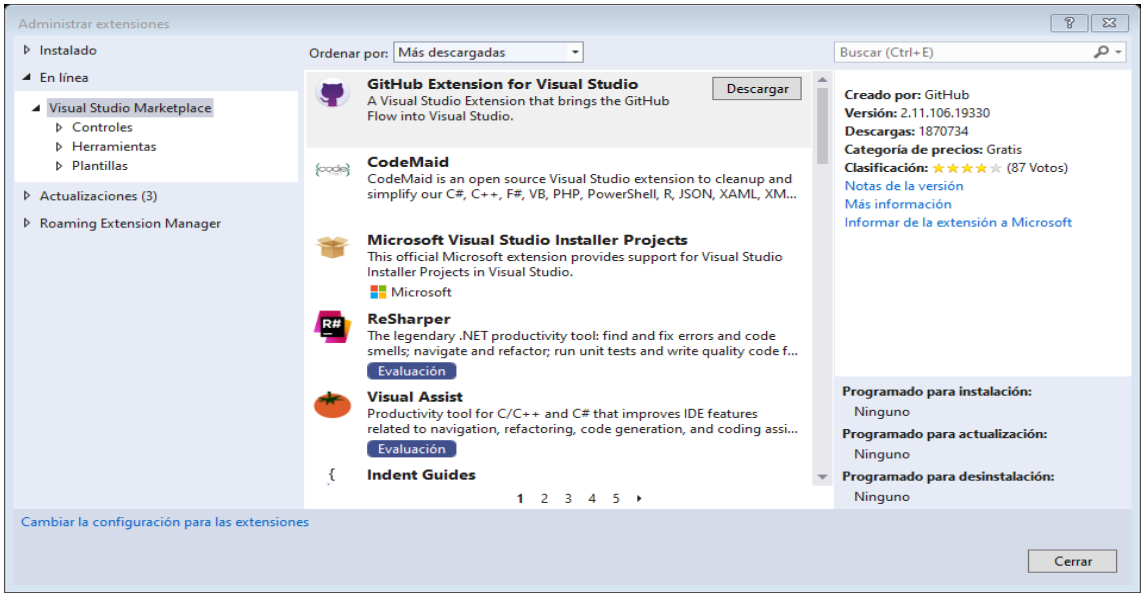
cuatro componentes de la pila de desarrollo LAMP y WAMP.

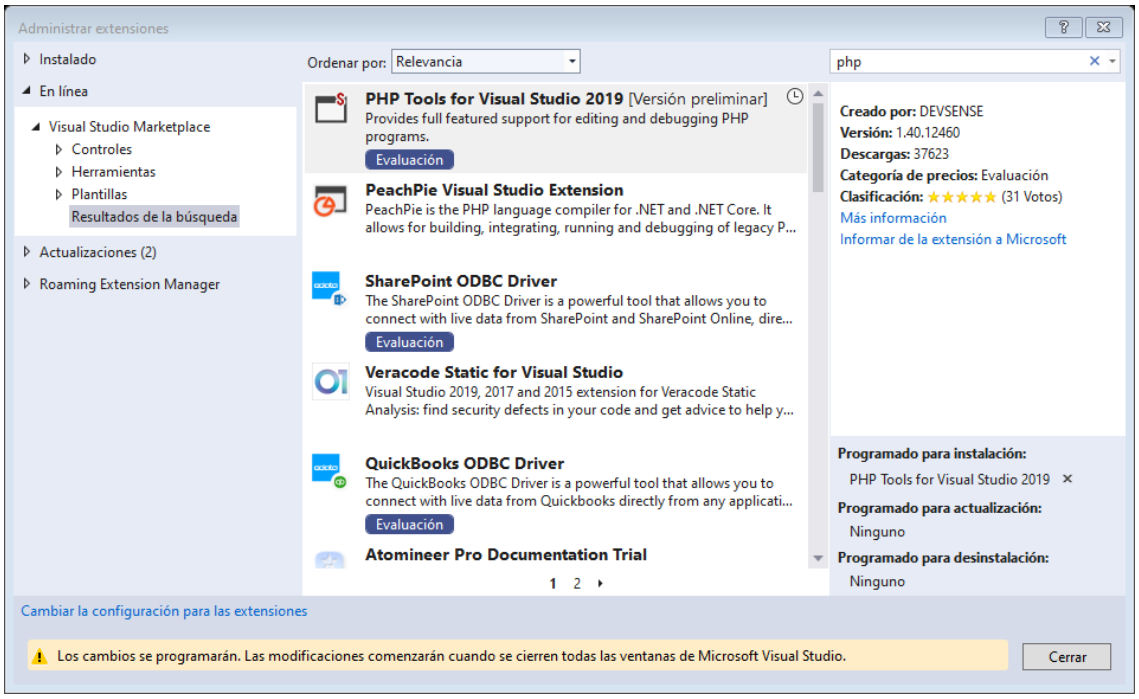
**IV**

**(La práctica tiene una duración de 2 horas) ACTIVIDADES**

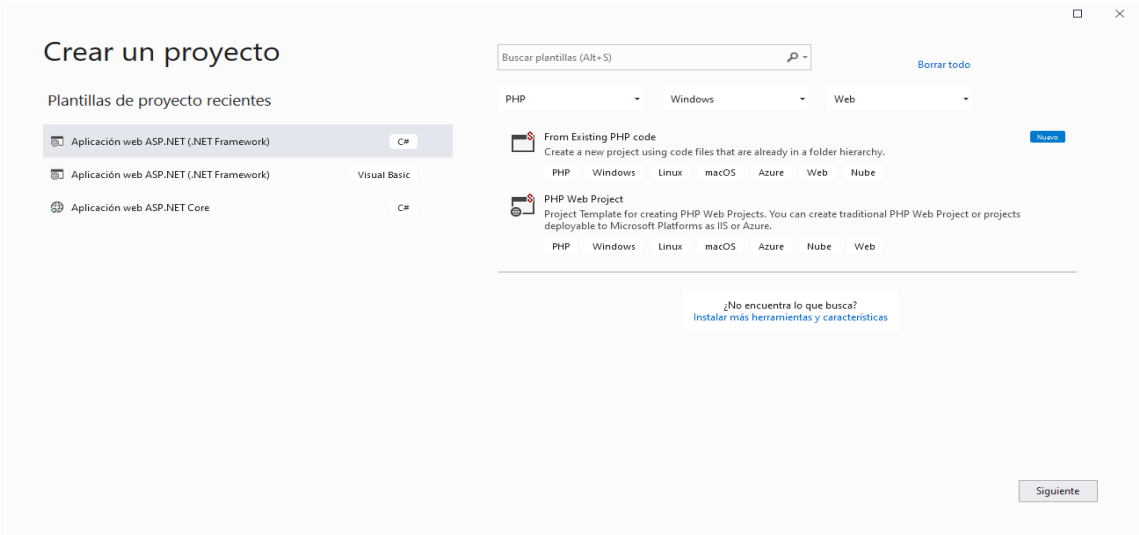
# 1. EXPERIENCIA DE PRÁCTICA N° 01: USAR PHP

1. Instalar PHP Tools, para ello ejecutar el Visual Studio 2019 y ejecutar Extensiones -> Administrar Extensiones, luego aparece la siguiente ventana:




2. Colocar PHPTools en Buscar, aparecerá la entrada hacer click en descargar, luego hacer la descarga y luego instalar la extensión.
3. Ahora crear un nuevo proyecto indicando que es en lenguaje PHP:




4. Probar los códigos proporcionados:

# B. FUNCIÓN DATE()

Devuelve una cadena formateada según el formato dado usando el parámetro de tipo integer `timestamp` dado o el momento actual si no se da una marca de tiempo.

| Carácter de format | Descripción | Ejemplo de valores devueltos |
|---|---|---|
| *Día* | --- | --- |
| *d* | Día del mes, 2 dígitos con ceros iniciales | *01* a *31* |
| *D* | Una representación textual de un día, tres letras | *Mon* hasta *Sun* |
| *j* | Día del mes sin ceros iniciales | *1* a *31* |
| *l* ('L' minúscula) | Una representación textual completa del día de la semana | *Sunday* hasta *Saturday* |
| *N* | Representación numérica ISO-8601 del día de la semana (añadido en PHP 5.1.0) | *1* (para lunes) hasta *7* (para domingo) |
| *S* | Sufijo ordinal inglés para el día del mes, 2 caracteres | *st*, *nd*, *rd* o *th*. Funciona bien con *j* |
| *w* | Representación numérica del día de la semana | *0* (para domingo) hasta *6* (para sábado) |
| *z* | El día del año (comenzando por 0) | *0* hasta *365* |
| *Semana* | --- | --- |
| *W* | Número de la semana del año ISO-8601, las semanas comienzan en lunes (añadido en PHP 4.1.0) | Ejemplo: *42* (la 42ª semana del año) |
| *Mes* | --- | --- |
| *F* | Una representación textual completa de un mes, como January o March | *January* hasta *December* |
| *m* | Representación numérica de una mes, con ceros iniciales | *01* hasta *12* |
| *M* | Una representación textual corta de un mes, tres letras | *Jan* hasta *Dec* |
| *n* | Representación numérica de un mes, sin ceros iniciales | *1* hasta *12* |
| *t* | Número de días del mes dado | *28* hasta *31* |
| *Año* | --- | --- |

| Carácter de format | Descripción | Ejemplo de valores devueltos |
|---|---|---|
| *L* | Si es un año bisiesto | *1* si es bisiesto, *0* si no. |
| *o* | Número de año ISO-8601. Esto tiene el mismo valor que *Y*, excepto que si el número de la semana ISO (*W*) pertenece al año anterior o siguiente, se usa ese año en su lugar. (añadido en PHP 5.1.0) | Ejemplos: *1999* o *2003* |
| *Y* | Una representación numérica completa de un año, 4 dígitos | Ejemplos: *1999* o *2003* |
| *y* | Una representación de dos dígitos de un año | Ejemplos: *99* o *03* |
| *Hora* | --- | --- |
| *a* | Ante meridiem y Post meridiem en minúsculas | *am* o *pm* |
| *A* | Ante meridiem y Post meridiem en mayúsculas | *AM* o *PM* |
| *B* | Hora Internet | *000* hasta *999* |
| *g* | Formato de 12 horas de una hora sin ceros iniciales | *1* hasta *12* |
| *G* | Formato de 24 horas de una hora sin ceros iniciales | *0* hasta *23* |
| *h* | Formato de 12 horas de una hora con ceros iniciales | *01* hasta *12* |
| *H* | Formato de 24 horas de una hora con ceros iniciales | *00* hasta *23* |
| *i* | Minutos, con ceros iniciales | *00* hasta *59* |
| *s* | Segundos, con ceros iniciales | *00* hasta *59* |
| *u* | Microsegundos (añadido en PHP 5.2.2). Observe que **date()** siempre generará *000000* ya que toma un parámetro de tipo integer, mientras que DateTime::format() admite microsegundos. | Ejemplo: *654321* |

1. Escriba el siguiente código:

```
<?php
    $dia=date("d");
    if ($dia>10)
    {
        echo "sitio activo";
    }
    else
    {
        echo "sitio fuera de servicio";
    }
?>
```

2. Ejecute en el navegador y verifique sus resultados

# C. VARIABLES

Los nombres de variables comienzan con el signo $ y son sensibles a mayúsculas y minúsculas (no así las palabras claves del lenguaje).

En PHP no es necesario definir el tipo antes de utilizarla, las mismas se crean en el momento de emplearlas. Las variables se declaran cuando se le asigna un valor, por ejemplo:

```
$dia = 24; //Se declara una variable de tipo integer.
$sueldo = 758.43; //Se declara una variable de tipo double.
$nombre = "juan"; //Se declara una variable de tipo string.
$exite = true; //Se declara una variable boolean.
```

También podemos hacer notar que para disponer comentarios de línea debemos utilizar dos caracteres //

1. Escriba el siguiente código:

```php
<?php
    $dia = 24;
    $sueldo = 758.43;
    $nombre = "juan";
    $exite = true;
    echo "Variable entera: ";
    echo $dia;
    echo "<br>";
    echo "Variable double: ";
    echo $sueldo;
    echo "<br>";
    echo "Variable string: ";
    echo $nombre;
    echo "<br>";
    echo "Variable boolean: ";
    echo $exite;
    echo "<br>";
?>
```

2. Ejecute en el navegador y verifique sus resultados

# D. SENTENCIAS CONDICIONALES

Cuando se pretende que el programa, una vez llegado a un cierto punto, tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución difieren, se utiliza el conjunto de instrucciones:

if, else y elseif. La estructura base de este tipo de instrucciones es la siguiente:

```
if (Condición)
{
  Instrucción 1;
  Instrucción 2;
}
else
{
  Instrucción A;
  Instrucción B;
}
```

1. Escriba el siguiente código:

```php
<?php
    $valor = rand(1,100);
    echo "El Valor sorteado es: $valor<br>";
    if($valor <=9)
    {
        echo "Tiene un digito";
    }
    else
    {
        if($valor < 100)
        {
            echo "Tiene 2 digitos";
        }
        else
        {
            echo "Tiene 3 digitos";
        }
    }
?>
```

2. Ejecute en el navegador y verifique sus resultados

# E. SENTENCIAS REPETITIVAS

Las estructuras repetitivas son similares al lenguaje C.

1. Escriba el siguiente código:

```php
<?php
for ($f =1; $f<=100; $f++)
{
    echo $f;
    echo "<br>";
}
?>
```

2. Ejecute en el navegador y verifique sus resultados

# F. FORMULARIOS (CONTROLES TEXT Y SUBMIT)

El proceso para el manejo de FORMULARIOS requiere generalmente dos páginas, una que implementa el formulario y otra que procesa los datos cargados en el formulario.

La estructura mínima de un formulario es la siguiente: para la entrada de un nombre de persona, un objeto text y un botón para el envío del dato al servidor:

```php
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Formulario de entrada de datos</title>
</head>
<body>
    <form method="post" action="pagina2.php">
          Ingrese su nombre:
          <input type="text" name="nombre" />
          <br />
          <input type="submit" value="confirmar" />
    </form>
</body>
</html>
```

```php
<?php
    echo "El nombre ingresado es: ";
    echo $_REQUEST['nombre'];
?>
```

# G. FORMULARIO (CONTROL RADIO)

1. Escriba el siguiente código:

```php
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Problema</title>
</head>
<body>
    <form action="pagina2.php" method="post">
          Ingrese primer valor:
          <input type="text" name="valor1" />
```

```
            <br /> <br />
            Ingrese segundo valor:
            <input type="text" name="valor2" />
            <br /> <br />
            <input type="radio" name="radio1" value="suma" />sumar
            <br />
            <input type="radio" name="radio1" value="resta" />restar
            <br />
            <input type="submit" value="Operar" />
    </form>
</body>
</html>
```

```
// pagina2.php:
<?php
if ($_REQUEST['radio1']=="suma")
{
    $suma = $_REQUEST['valor1'] + $_REQUEST['valor2'];
    echo "La suma es: ".$suma;
}
else
{
    if ($_REQUEST['radio1']=="resta")
    {
        $resta = $_REQUEST['valor1'] - $_REQUEST['valor2'];
        echo "La resta es: ".$resta;
    }
}
?>
```

2. En el editor de su preferencia, escriba en el archivo **index.html** el siguiente código

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <FORM METHOD="post" ACTION="mis_datos.php">
        <input type="hidden" name="edad" value="30">
        <p>Tu nombre <input type="text" name="nombre" size="30" value="paty"></p>
        <p>Tu edad <input type="text" name="edad" size="30" /></p>
        <p>Tu sistema favorito
        <select size="1" name="sistema">
            <option selected value="Linux">Linux</option>
            <option value="Unix">Unix</option>
            <option value="Macintosh">Macintosh</option>
            <option value="Windows">Windows</option>
            </select></p>
        <p>¿Te gusta el futbol?<input type="checkbox" name="futbol" value="ON"></p>
        <p>¿Cual es tu sexo?</p>
        <blockquote>
        <p>Hombre<input type="radio" value="hombre" checked name="sexo"></p>
        <p>Mujer <input type="radio" name="sexo" value="mujer"></p>
        </blockquote>
        <p>Aficiones</p>
        <p><textarea rows="5" name="aficiones" cols="28"></textarea></p>
        <p><input type="submit" value="Enviar datos" name="enviar">
        <input type="reset" value="Restablecer" name="B2"></p>
    </FORM>
```

```
</body>
</html>
```

3. En el archivo **mis_datos.php**, escriba el siguiente código:

```
<?php
// Recuperación de datos y almacenamiento en variables.
$nombre = $_POST['nombre'];
$edad = $_POST['edad'];
$sexo = $_POST['sexo'];
$sistema = $_POST['sistema'];
$futbol = $_POST['futbol'];
$aficiones = $_POST['aficiones'];
$edadsig = $edad + 1;
echo "Hola <b>". $nombre. "</b> que tal estás<BR>\n";
echo "Eres ". $sexo. "<BR>\n";
echo "El próximo año vas a cumplir ". $edadsig. " Años<BR>\n";
echo "Tu sistema favorito es ". $sistema. "<BR>\n";
if ($futbol)
{
    echo "Te gusta el futbol <BR>\n";
}
else
{
    echo "NO te gusta el futbol <BR>\n";
}
if ($aficiones != "")
{
    echo "Tus aficiones son: <BR>\n";
    echo $aficiones;
}
else
{
    echo "NO tienes aficiones <BR>\n";
}
?>
```

4. Ejecute en el navegador y verifique sus resultados
5. Escriba en el archivo **index.html** el siguiente código

```
<html><head></head><body>
<form method="post" action="enviar_mail.php">
 <table style="text-align: left; width: 100%;" cellpadding="3" rules="rows">
 <tbody>
   <tr>
     <td>Nombre:</td>
     <td><input type="text" name="nombre" id="nombre" /></td>
   </tr>
   <tr>
     <td>Correo electrónico:</td>
     <td><input type="text" name="email1" id="email1" /></td>
   </tr>
   <tr>
     <td>Repite tu correo:</td>
     <td><input type="text" name="email2" id="email2" /></td>
   </tr>
   <tr>
   <td>¿Cómo nos conociste?:</td>
     <td><select name="conocio" id="conocio">
      <option selected="selected"></option>
      <option>Un amigo</option>
      <option>Un blog</option>
      <option>Un buscador</option>
      <option>Otros</option>
```

```
    </select></td>
   </tr>
   <tr>
     <td>Tu pregunta es referente a:</td>
     <td><label><input type="radio" name="referente" id="ref_web" value="web" />
      La Web </label><br />
     <label><input type="radio" name="referente" id="ref_fotos" value="fotos" />
      Las fotos </label><br />
     <label><input      type="radio"      name="referente"      id="ref_asociacion"
value="asociacion" />
      La asocioaci&oacute;n </label></td>
   </tr>
   <tr>
     <td colspan="2">Escribe lo que quieres preguntarnos:</td>
   </tr>
   <tr>
     <td class="centrado" colspan="2">
      <textarea cols="50" rows="5" name="consulta"></textarea></td>
   </tr>
   <tr>
     <td class="centrado" colspan="2"><input type="submit" value="Enviar" />
       <input type="reset" value="Restablecer" /></td>
   </tr>
  </tbody>
 </table>
</form>
</body></html>
```

6. En el archivo **enviar_mail.php**, escriba el siguiente código:

```
<?php
//comenzamos recogiendo los datos
function recogeDato($campo){
 return isset($_REQUEST[$campo])?$_REQUEST[$campo]:'';
} //la función recogeDatos comprueba si se ha recibido un dato y recoge su valor

//si no se ha recibido, le asigna un valor vacío.
$email1    = recogeDato('email1');
$email2    = recogeDato('email2'); //asignamos cada valor a una variable
$consulta  = recogeDato('consulta');
$nombre    = recogeDato('nombre');
$conocio   = recogeDato('conocio');
$referente = recogeDato('referente');
$algunerror = FALSE;

//una vez recogidos, los validamos (campos obligatorios, etc...)
if($email1==''){ //validamos los que el email no esté vacio
 $algunerror = TRUE; //si encontramos un error, mostramos un mensaje
 echo "<p class=\"erroneo\">No has introducido tu eMail</p>\n";
} elseif($email1!=$email2){ //si tiene algo, que coincida con la repetición
 $algunerror = TRUE;
 echo "<p class=\"erroneo\">Los eMails introducidos no coinciden.</p>\n";
}
if($nombre==''){ //comprobamos que el nombre no haya quedado vacío
 $algunerror = TRUE;
 echo "<p class=\"erroneo\">No has introducido tu nombre.</p>\n";
}
if($consulta==''){ //comprobamos que el contenido de la pregunta no esté vacío
 $algunerror = TRUE;
 echo  "<p  class=\"erroneo\">El  área  de  la  consulta  no  puede  quedar  en
blanco.</p>\n";
}
if ($algunerror){ //comprobamos si ha habido algún error
 echo "<p> </p>\n"; //si los hay, se lo indicamos al usuario
```

```
 echo "<p>No se ha podido enviar el mensaje por los errores que se detallan
arriba.</p>\n";
 echo "<p>Por favor, vuelve a rellenar el formulario.</p>\n";
 echo      "<p      class=\"centrado\"><a      href=\"index.html\">Volver      al
formulario</a></p>\n";
}else{
 $para="ejemplo.aulaclic@gmail.com"; //si todo es correcto, enviamos el correo
 $asunto="Contacto web Flores - consulta sobre ".$referente;
 $mensaje="Datos del formulario de contacto:\n". //creamos el mensaje con los datos
 "Nombre: ".$nombre." \n".
 "eMail:  ".$email1."\n".
 "Nos conocio por: ".$conocio." \n".
 "Pregunta: ".$consulta;
 mail($para, $asunto, $mensaje); //y lo enviamos
 echo "<p>Tu mensaje se ha enviado correctamente. Gracias por contactar con
nosotros.</p>\n";
 echo "<p>Nos pondremos en contacto lo antes posible.</p>\n";
}
?>
```

7. Ejecute el código.

## Validación

8. Diseñe el siguiente formulario

**PHP Form Validation Example**

* required field.

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male *

Submit

**Your Input:**

9. Escriba en el siguiente código php

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = test_input($_POST["name"]);
    $email = test_input($_POST["email"]);
```

```
    $website = test_input($_POST["website"]);
    $comment = test_input($_POST["comment"]);
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
 $data = trim($data);
 $data = stripslashes($data);
 $data = htmlspecialchars($data);
 return $data;
}
?>
```

```
<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
 Name: <input type="text" name="name" >
 <br /><br />
 E-mail: <input type="text" name="email">
 <br /><br />
 Website: <input type="text" name="website">
 <br /><br />
 Comment: <textarea name="comment" rows="5" cols="40">
 </textarea>
 <br /><br />
 Gender:
 <input type="radio" name="gender" value="female">Female
 <input type="radio" name="gender" value="male">Male
 <br /><br />
 <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
```

10. Ejecute la aplicación
11. Modifique el código php anterior

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

```
 if (empty($_POST["name"])) {
  $nameErr = "Name is required";
 } else {
  $name = test_input($_POST["name"]);
  // check if name only contains letters and whitespace
  if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
   $nameErr = "Only letters and white space allowed";
  }
 }

 if (empty($_POST["email"])) {
  $emailErr = "Email is required";
 } else {
  $email = test_input($_POST["email"]);
  // check if e-mail address is well-formed
  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
   $emailErr = "Invalid email format";
  }
 }

 if (empty($_POST["website"])) {
  $website = "";
 } else {
  $website = test_input($_POST["website"]);
  // check if URL address syntax is valid (this regular expression also allows
dashes in the URL)
  if   (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-
a-z0-9+&@#\/%=~_|]/i",$website)) {
   $websiteErr = "Invalid URL";
  }
 }

 if (empty($_POST["comment"])) {
  $comment = "";
 } else {
  $comment = test_input($_POST["comment"]);
 }

 if (empty($_POST["gender"])) {
  $genderErr = "Gender is required";
 } else {
  $gender = test_input($_POST["gender"]);
 }
}

function test_input($data) {
 $data = trim($data);
 $data = stripslashes($data);
 $data = htmlspecialchars($data);
 return $data;
}
?>
```

```
<h2>PHP Form Validation Example</h2>
<p>
 <span class="error">* required field.</span>
</p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
 Name:
 <input type="text" name="name" value="<?php echo $name;?>" />
 <span class="error">
```

```
  * <?php echo $nameErr;?>
 </span>
 <br />
 <br />
 E-mail:
 <input type="text" name="email" value="<?php echo $email;?>" />
 <span class="error">
  * <?php echo $emailErr;?>
 </span>
 <br />
 <br />
 Website:
 <input type="text" name="website" value="<?php echo $website;?>" />
 <span class="error">
  <?php echo $websiteErr;?>
 </span>
 <br />
 <br />
 Comment:
 <textarea name="comment" rows="5" cols="40">
  <?php echo $comment;?>
 </textarea>
 <br />
 <br />
 Gender:
 <input type="radio" name="gender" <?php if (isset($gender) && $gender=="female")
echo "checked";?> value="female" />Female
 <input type="radio" name="gender" <?php if (isset($gender) && $gender=="male")
echo "checked";?> value="male" />Male
 <span class="error">
  * <?php echo $genderErr;?>
 </span>
 <br />
 <br />
 <input type="submit" name="submit" value="Submit" />
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
```

12. Ejecute la aplicación

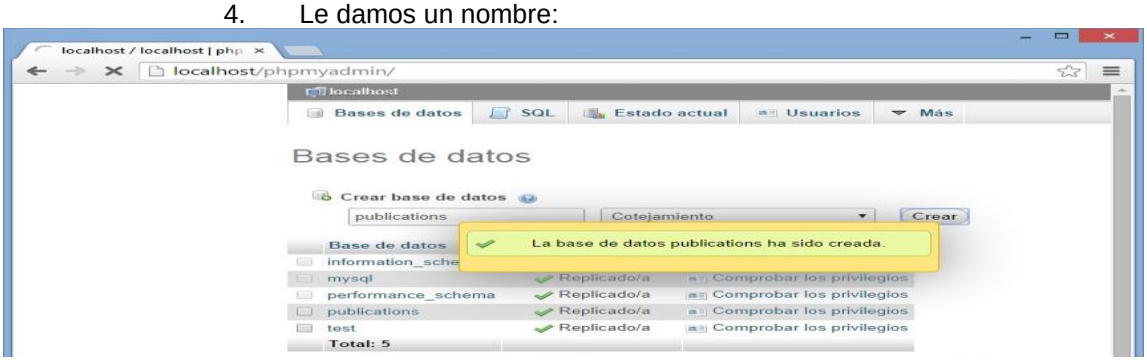
## 2. EXPERIENCIA DE PRÁCTICA N° 02: USAR A MYSQL

1. En el administrador de XAMP presione el botón Admin para MySQL, debe estar ejecutando el servidor web Apache

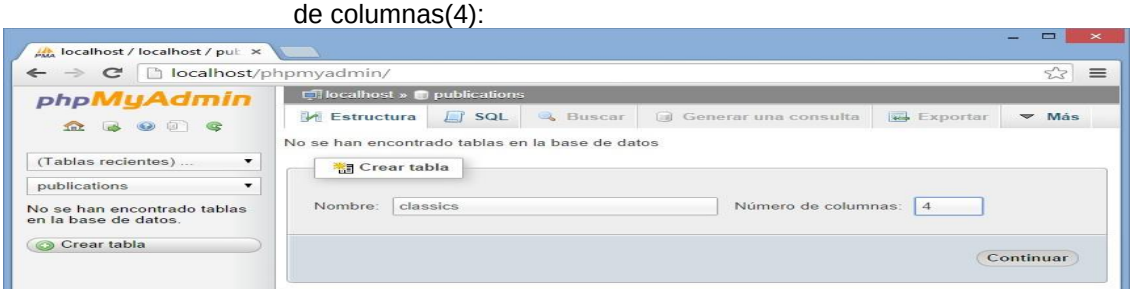

2. En la ventana phpmyadmin se elige la opción base de datos


3. Luego presionamos el botón crear:

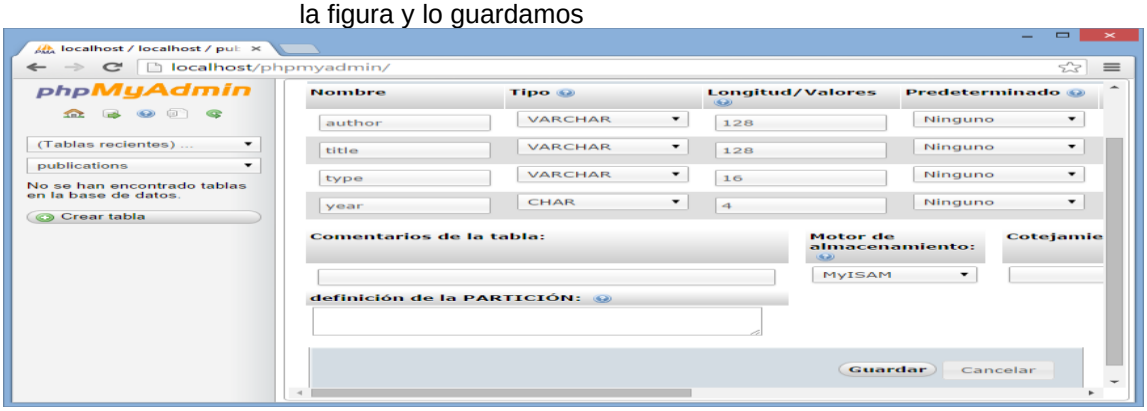

4. Le damos un nombre:




5. Creamos una nueva tabla, con el nombre classics indicando el número de columnas(4):

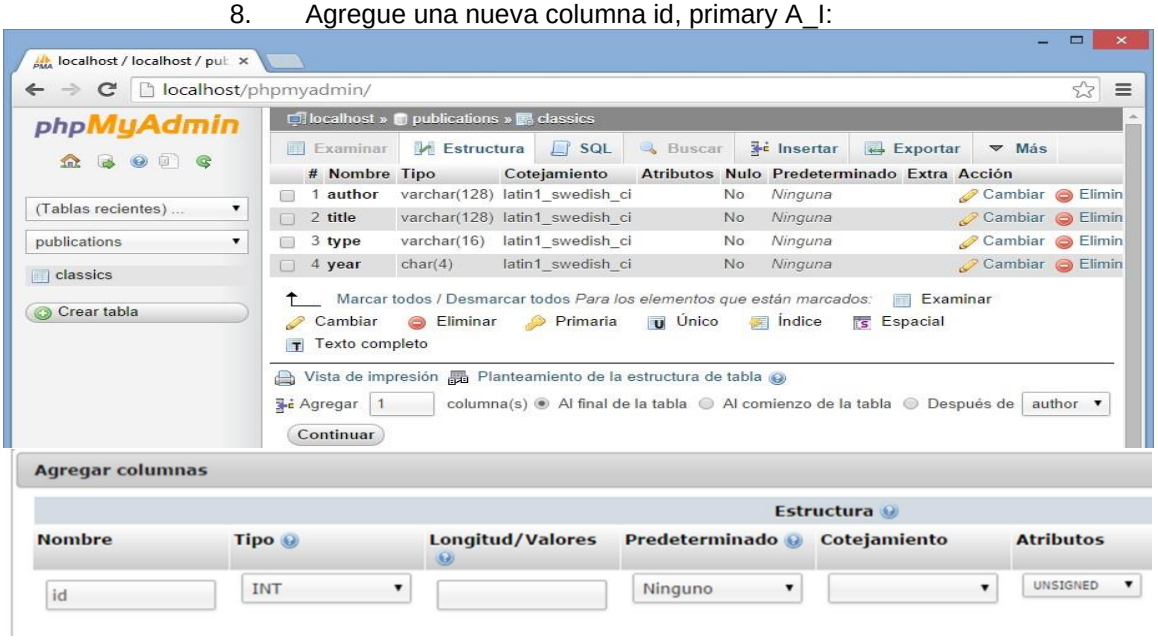



6. Definimos los campos con sus tipos, longitudes tal como se muestra en la figura y lo guardamos




7. Verifique la estructura

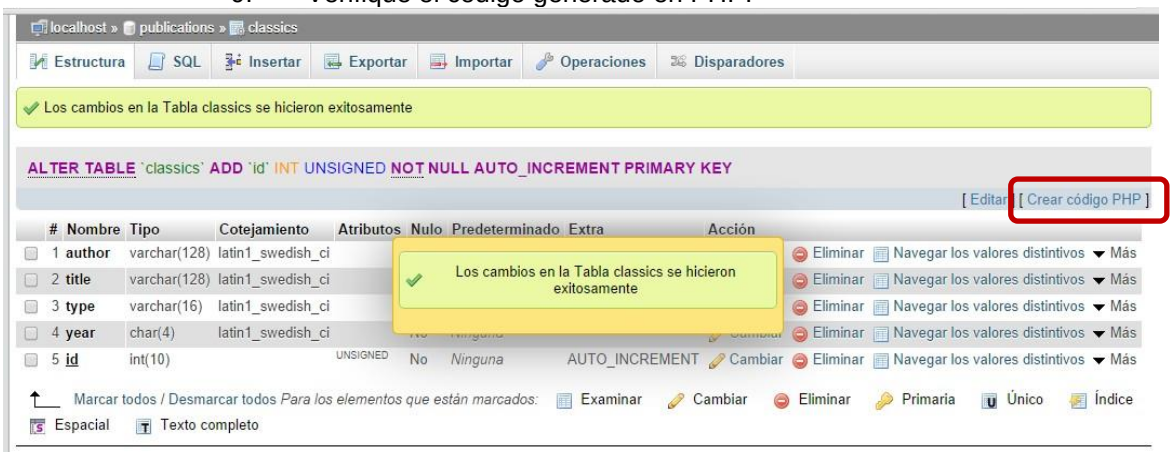

8. Agregue una nueva columna id, primary A_I:








9. Verifique el código generado en PHP:

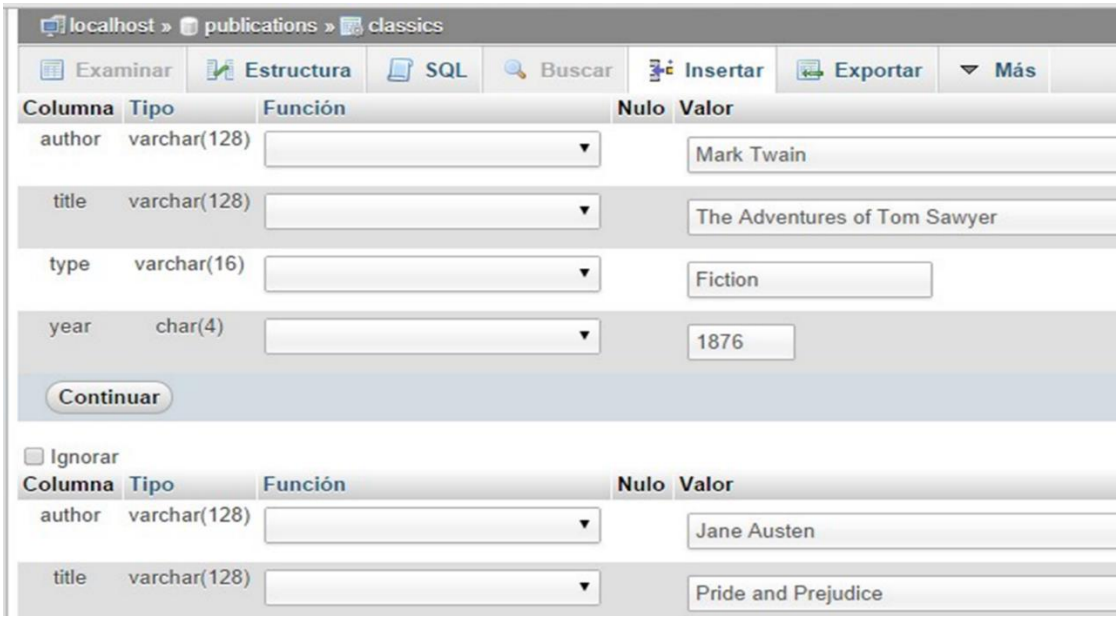



10. Muestre el código PHP

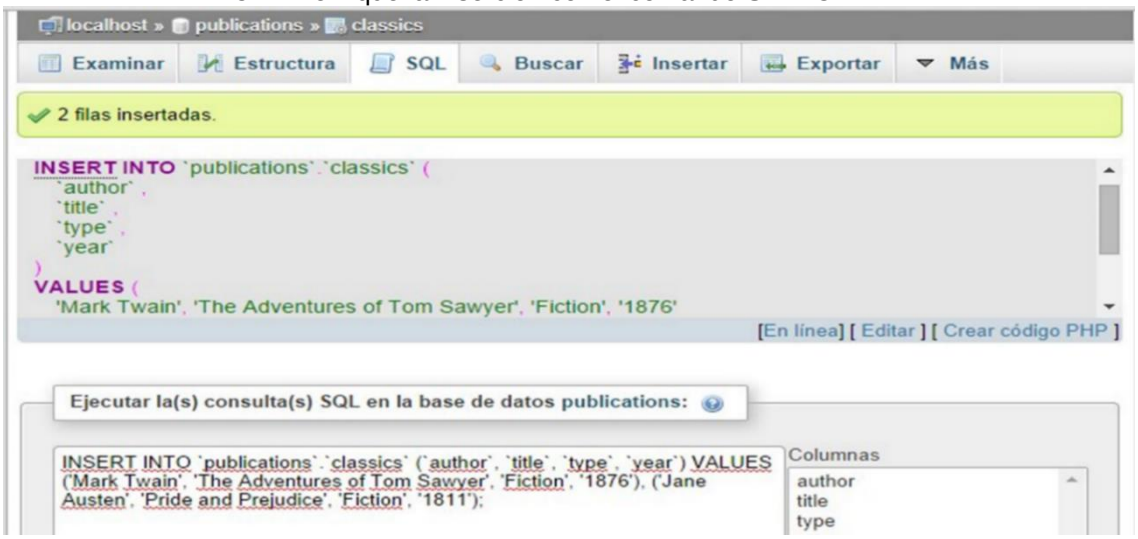

11. Elimine una columna:

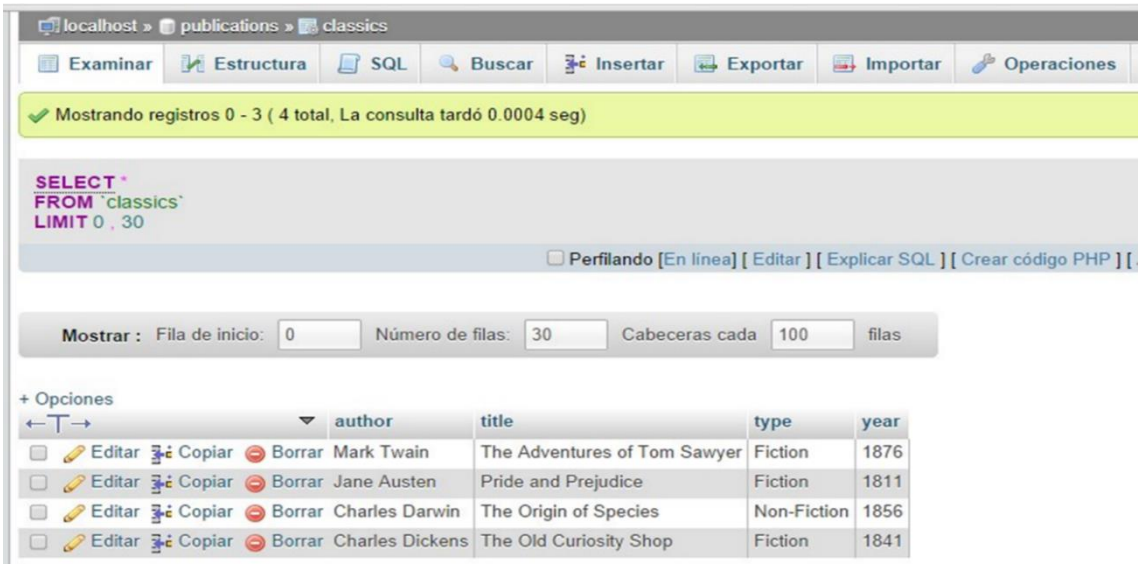

12. Inserte datos a la tabla con el comando INSERT:


```
INSERT INTO classics(author, title, type, year) VALUES('Mark Twain','The Adventures of Tom Sawyer','Fiction','1876');
INSERT INTO classics(author, title, type, year)
VALUES('Jane Austen','Pride and Prejudice','Fiction','1811'); INSERT INTO classics(author, title, type, year) VALUES('Charles Darwin','The Origin of Species','Non- Fiction','1856');
INSERT INTO classics(author, title, type, year) VALUES('Charles Dickens','The Old Curiosity Shop','Fiction','1841');
INSERT INTO classics(author, title, type, year) VALUES('William Shakespeare','Romeo and Juliet','Play','1594');
```




13. Verifique la inserción con el comando SELECT:

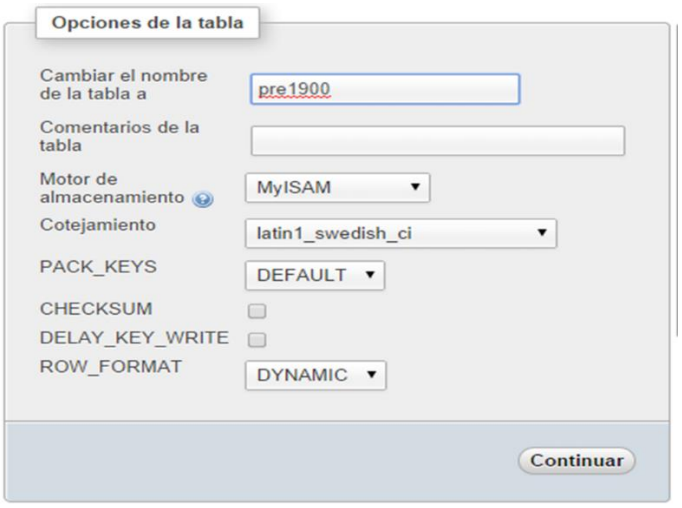



## Renombrar una tabla

14. Renombre la tabla classics por pre1900:

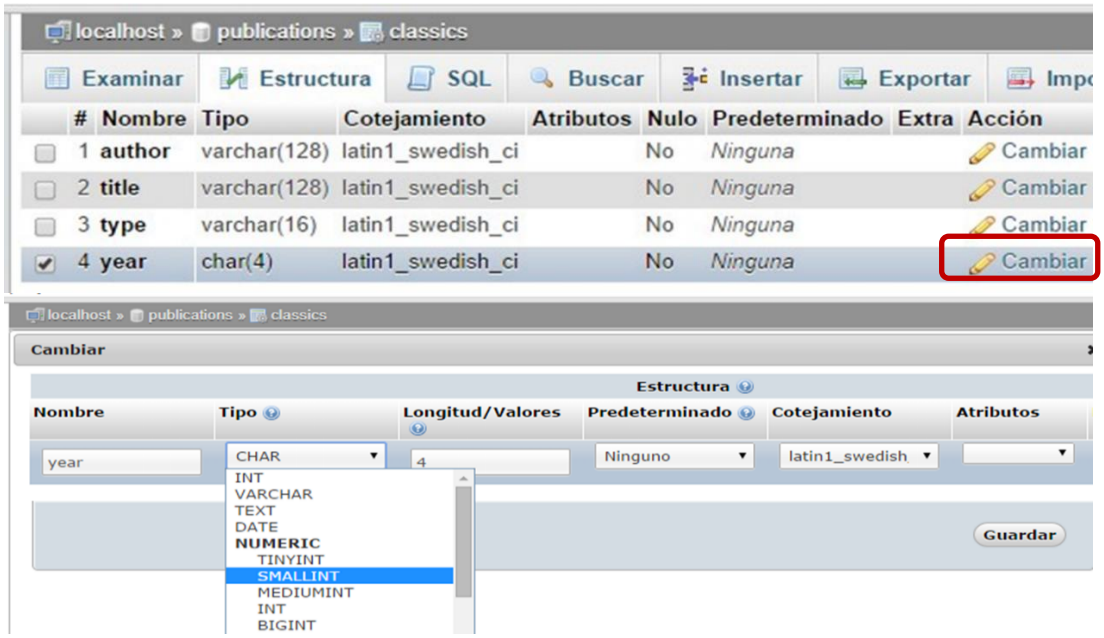
```
ALTER TABLE classics RENAME pre1900;
```




## Cambiar el tipo de dato de una columna

15. Cambie el tipo de dato de la columna year a smallint:

```
ALTER TABLE classics MODIFY year SMALLINT;
```




## Índices

16. Agregue los índices a la tabla classics:

```
ALTER TABLE classics ADD INDEX(author(20)); ALTER TABLE classics ADD INDEX(title(20)); ALTER TABLE classics ADD INDEX(category(4)); ALTER TABLE classics ADD INDEX(year); DESCRIBE classics;
```


# 3. EXPERIENCIA DE PRÁCTICA N° 02: USAR COMANDOS MYSQL

1. De forma predeterminada, el usuario inicial de MySQL es root y tendrá una contraseña predeterminada de mysql. Entonces, para ingresar a la interfaz de línea de comandos de MySQL, seleccione Inicio → Ejecutar, ingrese CMD en el cuadro Ejecutar y presione Retorno. Esto llamará un símbolo del sistema de Windows. Desde allí, ingrese lo siguiente (haciendo los cambios apropiados como se acaba de discutir):

**Ejemplo N° 9 - 1: Ejecutar MySQL**

```
cd C:\Archivos de programa (x86)\Xampps\mysql\bin
mysql –u root –p mysql
```

Nota: no hay punto y coma al final del comando anterior

El primer comando cambia al directorio MySQL y el segundo le dice a MySQL que inicie sesión como usuario root, con la contraseña mysql.

**Tabla N° 1: Las seis solicitudes de comando de MySQL**

| Promp de MySQL | Significado |
|---|---|
| mysql> | Listo y esperando un comando |
| -> | Esperando la siguiente línea de un comando |
| '> | Esperando la siguiente línea de una cadena comenzada con una comilla simple |
| "> | Esperando la siguiente línea de una cadena comenzada con comillas dobles |
| `> | Esperando la siguiente línea de una cadena comenzada con una tilde invertida |
| /*> | Esperando la siguiente línea de un comentario que comienza con / * |

Si está a medio camino de ingresar un comando y decide que no desea ejecutarlo después de todo, haga lo que haga, ¡no presione Ctrl-C! eso cerrará el programa. En su lugar, puede ingresar \c y presionar ENTER.

Los comandos SQL y las palabras clave no distinguen entre mayúsculas y minúsculas. CREATE, create, y CrEaTe significan lo mismo. Sin embargo, en aras de la claridad, es posible que prefiera utilizar mayúsculas.

Los nombres de las tablas distinguen entre mayúsculas y minúsculas en Linux y macOS, pero no distinguen entre mayúsculas y minúsculas en Windows. Por lo tanto, en aras de la portabilidad, siempre debe elegir un estuche y ceñirse a él. El estilo recomendado es usar minúsculas para los nombres de las tablas.

**Ejemplo N° 9 - 2: Cancelar una línea de entrada**

```
meaningless gibberish to mysql \c
```

**Tabla N° 2: Comandos comunes de MySQL**

| COMANDO | ACCIÓN |
|---|---|
| ALTER | Modificar una base de datos o una tabla |
| BACKUP | Hacer copia de seguridad de una tabla |
| \c | Cancelar entrada |
| CREATE | Crea una base de datos |
| DELETE | Eliminar una fila de una tabla |
| DESCRIBE | Describe las columnas de una tabla |
| DROP | Eliminar una base de datos o una tabla |
| EXIT (Ctrl-C) | Salir |
| GRANT | Cambiar privilegios de usuario |
| HELP (\h, \?) | Mostrar ayuda |
| INSERT | Insertar datos |
| LOCK | Bloquear tabla (s) |
| QUIT (\q) | Igual que SALIR |
| RENAME | Cambiar el nombre de una tabla |
| SHOW | Lista de detalles sobre un objeto |
| SOURCE | Ejecutar un archivo |
| STATUS (\s) | Muestra el estado actual |
| TRUNCATE | Vaciar una mesa |
| UNLOCK | Desbloquear tabla (s) |
| UPDATE | Actualizar un registro existente |
| USE | Utilizar una base de datos |

**Ejemplo N° 9 - 3: Ejemplo de creación de una tabla**

```
CREATE DATABASE publications;// crear una base de datos llamada "publications"
USE publications; // trabajando en la base de datos recién creada
```

**Ejemplo N° 9 - 4: Parámetros de ejemplo para el comando GRANT**

| Argumentos | Significado |
|---|---|
| *. * | Todas las bases de datos y todos sus objetos |
| base de datos. * | Solo la base de datos llamada base de datos y todos sus objetos |
| database.object | Solo la base de datos llamada database y su objeto llamado object |

```
GRANT ALL ON publications.* TO 'jim'@'localhost'
  IDENTIFIED BY 'mypasswd';
```

Lo que esto hace es permitir al usuario jim@localhost acceso completo a la base de datos de publicaciones usando la contraseña mypasswd. Puede probar si este paso ha funcionado ingresando quit para salir y luego volviendo a ejecutar MySQL como lo hizo antes, pero en lugar de ingresar -u root -p, escriba -u jim -p, o cualquier nombre de usuario que haya creado. Consulte la Tabla 8-5 para conocer el comando correcto para su sistema operativo. Modifíquelo según sea necesario si el programa cliente mysql está instalado en un directorio diferente de su sistema.

Tenga en cuenta que, si crea un nuevo usuario, pero no especifica una cláusula IDENTIFIED BY, el usuario no tendrá contraseña, una situación que es muy insegura y debe evitarse.

entrando quit para salir

**Ejemplo N° 9 - 5: Creando una mesa llamada classics**

```
CREATE TABLE classics (
 author VARCHAR(128),
```

```
 title VARCHAR(128),
 type VARCHAR(16),
 year CHAR(4)) ENGINE InnoDB;
```

Para comprobar si se ha creado su nueva tabla, escriba lo siguiente:

```
DESCRIBE classics;
```

El término VARCHAR significa VARiable length CHARacter string, y el comando toma un valor numérico que le dice a MySQL la longitud máxima permitida para una cadena almacenada en este campo.

Otra característica de las columnas de caracteres y texto, importante para el alcance web global actual, son los conjuntos de caracteres. Estos asignan valores binarios particulares a caracteres particulares. El juego de caracteres que usa para inglés es obviamente diferente del que usaría para ruso. Puede asignar el juego de caracteres a un carácter o columna de texto cuando lo crea.

Tenga en cuenta que, si alguna vez intenta asignar un valor de cadena más largo que la longitud permitida, se truncará a la longitud máxima declarada en la definición de la tabla.

El tipo de datos YEAR solo admite los años 0000 y 1901 a 2155. Esto se debe a que MySQL almacena el año en un solo byte por razones de eficiencia, pero significa que solo están disponibles 256 años.

Los tipos de datos BINARIOS almacenan cadenas de bytes que no tienen un juego de caracteres asociado. Por ejemplo, puede utilizar el tipo de datos BINARY para almacenar una imagen GIF.

MySQL indexa solo los primeros n caracteres de una columna TEXT (usted especifica n cuando crea el índice). Lo que esto significa es que VARCHAR es el tipo de datos mejor y más rápido para usar si necesita buscar todo el contenido de un campo.

El término BLOB significa Binar Large OBject y, por lo tanto, como podría pensar, el tipo de datos BLOB es más útil para datos binarios con un tamaño superior a 65,536 bytes. La otra diferencia principal entre los tipos de datos BLOB y BINARY es que los BLOB no pueden tener valores predeterminados.

Los valores de coma flotante (de cualquier precisión) solo se pueden firmar.

Los tipos de datos DATETIME y TIMESTAMP se muestran de la misma manera. La principal diferencia es que TIMESTAMP tiene un rango muy estrecho (desde los años 1970 hasta 2037), mientras que DATETIME tendrá casi cualquier fecha que probablemente especifique, a menos que esté interesado en historia antigua o ciencia ficción.

**Ejemplo N° 9 - 6: Agregar la columna id de incremento automático**

```
ALTER TABLE classics ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY;
```

**Ejemplo 9-7. Eliminar la columna de identificación**

```
ALTER TABLE classics DROP id;
```

**Ejemplo N° 9 - 7: Ingresando datos a la tabla classics**

```
INSERT INTO classics(author, title, type, year)
 VALUES('Mark Twain','The Adventures of Tom Sawyer','Fiction','1876');
INSERT INTO classics(author, title, type, year)
 VALUES('Jane Austen','Pride and Prejudice','Fiction','1811');
INSERT INTO classics(author, title, type, year)
 VALUES('Charles Darwin','The Origin of Species','Non-Fiction','1856');
INSERT INTO classics(author, title, type, year)
 VALUES('Charles Dickens','The Old Curiosity Shop','Fiction','1841');
INSERT INTO classics(author, title, type, year)
```

```
VALUES('William Shakespeare','Romeo and Juliet','Play','1594');
```

Mostrando el contenido de la tabla:

```
SELECT * FROM classics;
```

Cambiar el nombre de una tabla; para cambiar el nombre de los clásicos de la mesa a pre1900, usaría el siguiente comando:

```
ALTER TABLE classics RENAME pre1900;
```

Cambiar el tipo de datos de una columna:

```
ALTER TABLE classics MODIFY year SMALLINT;
```

Agregar una nueva columna:

```
ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
```

Cambiar el nombre de una columna:

```
ALTER TABLE classics CHANGE type category VARCHAR(16);
```

Eliminar una columna:

```
ALTER TABLE classics DROP pages;
```

**Ejemplo N° 9 - 8: Crear, ver y eliminar una tabla**

```
CREATE TABLE disposable(trash INT);
DESCRIBE disposable;
DROP TABLE disposable;
SHOW tables;
```

**Ejemplo N° 9 - 9: Agregar índices a la tabla classics**

```
ALTER TABLE classics ADD INDEX(author(20));
ALTER TABLE classics ADD INDEX(title(20));
ALTER TABLE classics ADD INDEX(category(4));
ALTER TABLE classics ADD INDEX(year);
DESCRIBE classics;
```

Una alternativa a usar ALTER TABLE para agregar un índice es usar el comando CREATE INDEX. Son equivalentes, excepto que CREATE INDEX no se puede utilizar para crear una CLAVE PRIMARIA.

**Ejemplo N° 9 - 10: Estos dos comandos son equivalentes**

```
ALTER TABLE classics ADD INDEX(author(20));
CREATE INDEX author ON classics (author(20));
```

**Ejemplo N° 9 - 11: Creando la tabla classics con índices**

```
CREATE TABLE classics (
 author VARCHAR(128),
 title VARCHAR(128),
 category VARCHAR(16),
 year SMALLINT,
 INDEX(author(20)),
 INDEX(title(20)),
 INDEX(category(4)),
 INDEX(year)) ENGINE InnoDB;
```

Eliminar una tabla:

```
DROP TABLE classics1;
```

Todos los valores deben ser únicos en cualquier columna que tenga un índice de clave principal.

**Ejemplo N° 9 - 12:Llenar la columna isbn con datos y usar una clave principal**

```
ALTER TABLE classics ADD isbn CHAR(13);
UPDATE classics SET isbn='9781598184891' WHERE year='1876';
UPDATE classics SET isbn='9780582506206' WHERE year='1811';
UPDATE classics SET isbn='9780517123201' WHERE year='1856';
UPDATE classics SET isbn='9780099533474' WHERE year='1841';
UPDATE classics SET isbn='9780192814968' WHERE year='1594';
ALTER TABLE classics ADD PRIMARY KEY(isbn);
DESCRIBE classics;
```

**Ejemplo N° 9 - 13: Creando la tabla classics con una clave primaria**

```
CREATE TABLE classics (
 author VARCHAR(128),
 title VARCHAR(128),
 category VARCHAR(16),
 year SMALLINT,
 isbn CHAR(13),
 INDEX(author(20)),
 INDEX(title(20)),
 INDEX(category(4)),
 INDEX(year),
 PRIMARY KEY (isbn)) ENGINE InnoDB;
```

**Ejemplo N° 9 - 14: Agregar un índice FULLTEXT a la tabla classics**

```
ALTER TABLE classics ADD FULLTEXT(author,title);
```

**Ejemplo N° 8 - 15: Dos declaraciones SELECT diferentes**

```
SELECT author,title FROM classics;
SELECT title,isbn FROM classics;
```

**Ejemplo N° 9 - 16: Contando filas**

```
SELECT COUNT(*) FROM classics;
```

**Ejemplo N° 9 - 17: Con y sin el calificador DISTINCT**

```
SELECT author FROM classics;
```

```
SELECT DISTINCT author FROM classics;
```

**Ejemplo N° 9 - 18: Eliminar la nueva entrada**

```
DELETE FROM classics WHERE title='Little Dorrit';
```

**Ejemplo N° 9 - 19: Usando la palabra clave WHERE**

```
SELECT author,title FROM classics WHERE author="Mark Twain";
SELECT author,title FROM classics WHERE isbn="9781598184891";
```

**Ejemplo N° 9 - 20: Usando el calificador LIKE**

```
SELECT author,title FROM classics WHERE author LIKE "Charles%";
SELECT author,title FROM classics WHERE title LIKE "%Species";
SELECT author,title FROM classics WHERE title LIKE "%and%";
```

**Ejemplo N° 9 - 21: Limitar el número de resultados devueltos**

```
SELECT author,title FROM classics LIMIT 3;
SELECT author,title FROM classics LIMIT 1,2;
SELECT author,title FROM classics LIMIT 3,1;
```

**Ejemplo N° 9 - 22: Usando MATCH ... AGAINST en índices FULLTEXT**

```
SELECT author,title FROM classics
 WHERE MATCH(author,title) AGAINST('and');
SELECT author,title FROM classics
 WHERE MATCH(author,title) AGAINST('curiosity shop');
SELECT author,title FROM classics
 WHERE MATCH(author,title) AGAINST('tom sawyer');
```

**Ejemplo N° 9 - 23: Usando MATCH ... AGAINST en modo booleano**

```
SELECT author,title FROM classics
 WHERE MATCH(author,title)
 AGAINST('+charles -species' IN BOOLEAN MODE);
SELECT author,title FROM classics
 WHERE MATCH(author,title)
 AGAINST('"origin of"' IN BOOLEAN MODE);
```

**Ejemplo N° 9 - 24: Usando UPDATE ... SET**

```
UPDATE classics SET author='Mark Twain (Samuel Langhorne Clemens)'
 WHERE author='Mark Twain';
UPDATE classics SET category='Classic Fiction'
 WHERE category='Fiction';
```

**Ejemplo N° 9 - 25: Usando ORDER BY**

```
SELECT author,title FROM classics ORDER BY author;
SELECT author,title FROM classics ORDER BY title DESC;
SELECT author,title,year FROM classics ORDER BY author,year DESC;
```

```
SELECT category,COUNT(author) FROM classics GROUP BY category;
```

**Ejemplo N° 9 - 26: Creando y llenando la tabla de clientes**

Clientes

```
CREATE TABLE customers (
 name VARCHAR(128),
 isbn VARCHAR(13),
 PRIMARY KEY (isbn)) ENGINE InnoDB;
INSERT INTO customers(name,isbn)
 VALUES('Joe Bloggs','9780099533474');
INSERT INTO customers(name,isbn)
 VALUES('Mary Smith','9780582506206');
INSERT INTO customers(name,isbn)
 VALUES('Jack Wilson','9780517123201');
SELECT * FROM customers;
```

**Ejemplo N° 9 - 27: Uniendo dos tablas en un solo SELECT**

```
SELECT name,author,title FROM customers,classics
 WHERE customers.isbn=classics.isbn;

 NATURAL JOIN toma dos tablas y une automáticamente columnas que tienen el mismo
nombre.

SELECT name,author,title FROM customers NATURAL JOIN classics;
```

```
 SELECT name,author,title FROM customers
 JOIN classics ON customers.isbn=classics.isbn;
```

También puede ahorrarse algo de escritura y mejorar la legibilidad de la consulta creando alias utilizando la palabra clave AS. Simplemente siga el nombre de una tabla con AS y el alias a usar.

```
SELECT name,author,title from
customers AS cust, classics AS class WHERE cust.isbn=class.isbn;
```

También puede usar AS para cambiar el nombre de una columna (si se une a tablas o no), así:

```
SELECT name AS customer FROM customers ORDER BY customer;
```

**Ejemplo N° 8 - 28: Usando operadores lógicos**

```
SELECT author,title FROM classics WHERE
 author LIKE "Charles%" AND author LIKE "%Darwin";
SELECT author,title FROM classics WHERE
 author LIKE "%Mark Twain%" OR author LIKE "%Samuel Langhorne Clemens%";
SELECT author,title FROM classics WHERE
 author LIKE "Charles%" AND author NOT LIKE "%Darwin";
```

**Ejemplo N° 9 - 29: Crear una tabla lista para transacciones**

```
CREATE TABLE accounts (
 number INT, balance FLOAT, PRIMARY KEY(number)
 ) ENGINE InnoDB;
```

```
DESCRIBE accounts;
```

**Ejemplo N° 9 - 30: Llenando la tabla de cuentas**

```
INSERT INTO accounts(number, balance) VALUES(12345, 1025.50);
INSERT INTO accounts(number, balance) VALUES(67890, 140.00);
SELECT * FROM accounts;
```

**Ejemplo N° 9 - 31: Una transacción MySQL**

```
BEGIN;
UPDATE accounts SET balance=balance+25.11 WHERE number=12345;
COMMIT;
SELECT * FROM accounts;
```

Cuando esté satisfecho de que una serie de consultas en una transacción se completó con éxito, emita un comando COMMIT para confirmar todos los cambios en la base de datos. Hasta que reciba un COMMIT, MySQL considera que todos los cambios que realiza son meramente temporales. Esta función le brinda la oportunidad de cancelar una transacción al no enviar un COMMIT, sino al emitir un comando ROLLBACK.

**Ejemplo N° 9 - 32: Una transacción de transferencia de fondos**

```
BEGIN;
UPDATE accounts SET balance=balance-250 WHERE number=12345;
UPDATE accounts SET balance=balance+250 WHERE number=67890;
SELECT * FROM accounts;
```

**Ejemplo N° 9 - 33: Cancelar una transacción usando ROLLBACK**

```
ROLLBACK;
SELECT * FROM accounts;
```

**Ejemplo N° 9 - 34: Usando el comando EXPLAIN**

```
EXPLAIN SELECT * FROM accounts WHERE number='12345';
```

Alternativamente, puede bloquear las tablas que está respaldando antes de ejecutar mysqldump. Para bloquear tablas para lectura (ya que queremos leer los datos), desde la línea de comandos de MySQL emita este comando:

```
LOCK TABLES tablename1 READ, tablename2 READ ...
Then, to release the lock(s), enter the following:
UNLOCK TABLES;
```

salga de MySQL usando:

```
quit;
```

El formato básico del comando mysqldump se muestra aquí:

```
mysqldump –u usuario –ppassword base de datos
```

Nota: sin punto y coma al final

**Ejemplo 9-8. Volcado de la base de datos de publicaciones en un archivo**

```
mysqldump -u user -ppassword database
```

**Copia de seguridad de una sola tabla**

Para hacer una copia de seguridad de una sola tabla de una base de datos (como la tabla *classics* de la base de datos de `publications`), primero debe bloquear la tabla desde la línea de comandos de MySQL, emitiendo un comando como el siguiente:

```
LOCK TABLES publications.classics READ;
```

Esto asegura que MySQL siga ejecutándose para fines de lectura,

pero no se puede escribir. Luego, mientras mantiene abierta la línea de comandos de MySQL, use otra ventana de terminal para ejecutar el siguiente comando desde la línea de comandos del sistema operativo:

```
mysqldump -u user -ppassword publications classics > classics.sql
```

Ahora debe liberar el bloqueo de la tabla ingresando el siguiente comando desde la línea de comandos de MySQL en la primera ventana de la terminal, que desbloquea todas las tablas que se han bloqueado durante la sesión actual:

```
UNLOCK TABLES;
```

**Ejemplo N° 9 - 35: Volcando todas las bases de datos MySQL al archivo**

```
mysqldump -u user -ppassword --all-databases > all_databases.sql
```

Restaurar desde un archivo de respaldo

Para realizar una restauración desde un archivo, llame al ejecutable mysql, pasándole el archivo para restaurar usando el símbolo <. Por lo tanto, para recuperar una base de datos completa que descargó usando la opción --all -bases, use un comando como el del Ejemplo 9-10.

**Ejemplo N° 9 - 36: Restaurar un conjunto completo de bases de datos**

```
mysql -u user -ppassword < all_databases.sql
```

Para restaurar una sola base de datos, use la opción -D seguida del nombre de la base de datos, como en el Ejemplo 9-11, donde la base de datos de publicaciones se está restaurando a partir de la copia de seguridad realizada en el Ejemplo 9-8.

**Ejemplo N° 9 - 37: Restaurar la base de datos de publicaciones**

```
mysql -u user -ppassword -D publications < publications.sql
```

Para restaurar una sola tabla en una base de datos, use un comando como el de donde solo se restaura la tabla de clásicos en la base de datos de publicaciones.

**Ejemplo 9-12. Restaurar la tabla de clásicos a la base de datos de publicaciones**

```
mysql -u user -ppassword -D publications < classics.sql
```

**Ejemplo N° 9 - 38: Volcado de datos en archivos en formato CSV**

```
mysqldump -u user -ppassword --no-create-info --tab=c:/temp
  --fields-terminated-by=',' publications
```

```
<?php // login.php
  $hn = 'localhost';
  $db = 'publications';
  $un = 'username'; // Change this
  $pw = 'password'; // Change this
?>
```

**Ejemplo N° 9 - 39: Conectarse a un servidor MySQL con mysqli**

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die("Fatal Error");
?>
```

El operador -> indica que el elemento de la derecha es una propiedad o método del objeto de la izquierda.

cada fila se puede recuperar en su totalidad a través del método fetch_array

heredoc (o here-document) es una forma de citar grandes cantidades de textos en shells y lenguajes de programación. Conserva los saltos de línea y otros formatos del texto literal; algunas implementaciones permiten interpolar variables, ejecutar el texto entre comillas como comandos o eliminar las pestañas iniciales. La sintaxis general es << seguida de un identificador, seguido, comenzando en la línea siguiente, por el texto que se va a citar, y luego cerrado por el mismo identificador en su propia línea. Tanto bourne shell como bash tienen heredocs como una forma de proporcionar entrada a los comandos. PHP y Perl también tienen implementaciones de heredocs.

**Ejemplo N° 9 - 40: Creando una mesa llamada gatos**

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die("Fatal Error");

  $query = "CREATE TABLE cats (
    id SMALLINT NOT NULL AUTO_INCREMENT,
    family VARCHAR(32) NOT NULL,
    name VARCHAR(32) NOT NULL,
    age TINYINT NOT NULL,
    PRIMARY KEY (id)
  )";

  $result = $conn->query($query);
  if (!$result) die ("Database access failed");
?>
```

**Ejemplo N° 9 - 41: Cómo acceder de forma segura a MySQL con la entrada del usuario**

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die("Fatal Error");

  $user  = mysql_fix_string($conn, $_POST['user']);
  $pass  = mysql_fix_string($conn, $_POST['pass']);
  $query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
```

```
  // Etc.

  function mysql_fix_string($conn, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
  }
?>
```

Un objeto grande binario (o BLOB) es una colección de datos binarios almacenados como una sola entidad en un sistema de administración de bases de datos. Los BLOB suelen ser imágenes, audio u otros objetos multimedia, aunque a veces el código binario se almacena como un BLOB.

**Ejemplo N° 9 - 42: Emitir declaraciones preparadas**

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die("Fatal Error");

  $stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?)');
  $stmt->bind_param('sssss', $author, $title, $category, $year, $isbn);

  $author   = 'Emily Brontë';
  $title    = 'Wuthering Heights';
  $category = 'Classic Fiction';
  $year     = '1847';
  $isbn     = '9780553212587';

  $stmt->execute();
  printf("%d Row inserted.\n", $stmt->affected_rows);
  $stmt->close();
  $conn->close();
?>
```

**Ejemplo N° 9 - 43: Funciones para prevenir ataques de inyección SQL y XSS**

```
<?php
  function mysql_entities_fix_string($conn, $string)
  {
    return htmlentities(mysql_fix_string($conn, $string));
  }

  function mysql_fix_string($conn, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
  }
?>
```

**Ejemplo N° 9 - 44: Enviar varios valores con una matriz**

```
Vanilla <input type="checkbox" name="ice[]" value="Vanilla">
 Chocolate <input type="checkbox" name="ice[]" value="Chocolate">
Strawberry <input type="checkbox" name="ice[]" value="Strawberry">
```

Las variables de sesión se almacenan en el servidor (a diferencia de las cookies, que se almacenan en el navegador web) y, por lo tanto, se puede confiar en ellas.

En lugar de la matriz $ _POST, los formularios se pueden cambiar fácilmente para usar el método GET. Las razones para hacer esto pueden incluir hacer que el resultado de una búsqueda se pueda marcar o vincular directamente desde otra página.

Alt-Flecha izquierda y Alt-Flecha derecha se mueven hacia atrás y hacia adelante dentro del

historial de navegación.

## V

## EJERCICIOS PROPUESTOS

1. Desarrolle una aplicación de venta de pasajes aéreos donde según la fecha de nacimiento permita determinar si es mayor o menor de edad determinando los precios a considerar:
   a) Adulto: precio completo
   b) Menores de edad (2 a 17 años): 75% del precio
   c) Infantes (hasta los 2 años): no pagan
2. Almacene aleatoriamente una matriz cuadrada y determine si dicha matriz es “perfecta”
3. Desarrolle una aplicación web para la inscripción de usuarios, según el formato adjunto y muestre una página de confirmación del registro de usuario.

**Formulario de inscripción de usuarios**

| | |
|---|---|
| Nombre completo | |
| Dirección | |
| Correo electrónico | |
| Contraseña | |
| Confirmar contraseña | |
| Fecha de nacimiento | Enero 1 (yyyy) |
| Sexo | Hombre Mujer |
| Por favor, elige los temas de interés | Ficción Terror Acción Comedia Suspense |
| Selecciona tus aficiones | Deporte-aire-libre |

**Página de confirmación del registro de usuario**

| | |
|---|---|
| **Nombre completo** | Antonio Becerra |
| **Dirección** | Universidad de Almería<br>Dpto. Lenguajes y Computación |
| **Correo electrónico** | abecerra@ual.es |
| **Contraseña** | antonio |
| **Fecha de nacimiento** | 15/Agosto/1970 |
| **Sexo** | Hombre |
| **Por favor, elige los temas de interés** | terror<br>accion<br>comedia |
| **Aficiones seleccionadas** | Musica-Pop<br>Fotografia |
| Confirmar datos | |

4. Desarrolle una aplicación para permita la carga de archivos por especialidad de la carrera y que se guarden en carpetas como: Estadistica, DesarrolloWeb, Testing, etc. y luego podamos ver el contenido de algún archivo seleccionado.

## V
## CUESTIONARIO

1. ¿Qué es MySQL?
2. ¿Qué es y para qué sirve MySQL?
3. ¿Qué es el programa MySQL?
4. ¿Cuáles son las ventajas y desventajas de MySQL?
5. ¿Cuáles son las versiones que existen de MySQL?
6. ¿Cómo se ejecuta MySQL?
7. ¿Cuáles son las principales operaciones de Bases de Datos que se realizan en MySQL?
8. ¿Cómo se define el tipo de dato de un campo en MySQL?
9. ¿Cómo se define la clave primaria en una tabla MySQL?
10. ¿Qué es PHP?
11. ¿De qué componentes consta WAMP?
12. ¿Cómo se construye una aplicación web con PHP?
13. ¿Qué son formularios web?
14. ¿Cuál es la diferencia entre las funciones include() y require()?
15. ¿Cómo podemos obtener la dirección IP del cliente?
16. ¿Cuál es la diferencia entre unset() y unlink()?
17. ¿Cuáles son los principales tipos de error en PHP y en qué se diferencian?
18. ¿Cuál es la diferencia entre GET y POST?
19. ¿Cómo se puede habilitar el reporte de errores en PHP?
20. ¿Qué son los Traits?
21. ¿Puede el valor de un cambio constante durante la ejecución del script?
22. ¿Se puede ampliar una clase final definida?
23. ¿Cuáles son los métodos __construct() y __destruct() en una clase PHP?
24. ¿Cómo podemos obtener el número de elementos de un array?
25. ¿Cómo declararías una función que recibe un nombre de parámetro 'hola'?
26. ¿Cómo funciona las aplicaciones web con formularios y PHP?

## VI
## BIBLIOGRAFIA Y REFERENCIAS


- Microsoft Official Course, Developing Web Applications using Microsoft Visual Studio 2010, Microsoft Coorporation, 2010
- Sánchez Flores, “Desarrollado aplicaciones con Visual C# NET 2008”. Ed. Macro, Perú, 2008
- Joseph Albabari y Ben Albarari, C# 4.0 in a Nutshell, O’Reilly Media., 2010
- Matthew MacDonald, “Beginning ASP.NET 2.0 in C# 2005”, Ed. Apress, USA, 2006.