MATERIA ELECTIVA III BASES DE DATOS WEB
ESCUELA: ANÁLISIS DE SISTEMAS
ÚLTIMA ACTUALIZACIÓN: FEBRERO 2019

Objetivo General:
Adquirir los conocimientos y competencias generales, que permitan tener una noción sobre las herramientas para el manejo de base de datos a través de soluciones tecnológicas basadas entornos web.

CONTENIDO

UNIDAD 1 PHP

1.1.- Primeros Pasos en PHP

HP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.
Hay que entender primero como funciona la solicitud de páginas en un navegador para comenzar a programar en PHP.

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] ?>
<html>
<head></head>
<body>
<?php
 echo "Hola Mundo";
?>
</body>
</html>
```

El comando de PHP para imprimir dentro de la página se llama echo.
Ejercicio de un problema sencillo que se nos puede presentar y que no se puede resolver empleando solo HTML es que una página esté disponible sólo los 10 primeros días del mes.
Mostraremos un cartel que diga que el sitio se encuentra disponible si la fecha es menor o igual a 10, en caso contrario mostraremos un mensaje de sitio fuera de servicio.
Para obtener la fecha del servidor web debemos llamar a la función date y requerir sólo el día:

```
$dia=date("d");
```

A las variables en PHP se les antecede el caracter $. Si a la función date le pasamos el string "d" retornará sólo el día (si queremos la fecha completa: $fecha=date("d/m/Y")
Para verificar si la variable $dia es menor o igual a 10, debemos emplear la instrucción if, similar a otros lenguajes.
Entonces la página con el programa queda de la siguiente forma:

```
<html>
<head></head>
<body>
<?php
 $dia=date("d");
 if ($dia<=10)
 {
  echo "sitio activo";
 }
 else
 {
```

```
   echo "sitio fuera de servicio";
  }
  ?>
</body>
</html>
```

Los nombres de variables son sensibles a mayúsculas y minúsculas, por lo que si la escribimos en minúscula inicialmente debemos respetar en el resto del programa. En cambio las instrucciones del lenguaje PHP no son sensibles por lo que si deseamos escribir IF o if, las dos formas estarán bien (es más común escribir las palabras claves siempre en minúsculas)
La condición del if debe ir obligatoriamente entre paréntesis.

Si la condición se verifica verdadera se ejecuta el primer bloque encerrado entre llaves, en caso de verificarse falsa la condición se ejecuta el bloque entre llaves que le sigue al else.

1.2.- 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 de dato que almacena 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.
```

Ejemplo
Hará la impresión de variables utilizaremos inicialmente el comando echo. Un programa completo que inicializa y muestra el contenido de cuatro variables de distinto tipo es:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
$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.
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;
?>
</body>
</html>
```

Ejemplo Variables tipo String
Una variable de este tipo puede almacenar una serie de caracteres.

```
$cadena1="Hola";
$cadena2="Mundo";
echo $cadena1." ".$cadena2;
```

Para concatenar string empleamos el operador . (punto)
Tengamos en cuenta que el comando echo de más arriba lo podemos hacer más largo de la siguiente forma:

```
echo $cadena1;
echo " ";
echo $cadena2;
```

A medida que uno haga ejercicios podremos resumir en un solo comando echo la salida de múltiples variables.
Cuando una cadena encerrada entre comillas dobles contiene una variable en su interior, ésta se trata como tal, por lo tanto se utilizará su contenido para el almacenamiento:

```
$dia=10;
$fecha="Hoy es $dia";
echo $fecha;
```

En pantalla se muestra: Hoy es 10
Es decir, en la cadena, se sustituye el nombre de la variable $dia, con el contenido de la misma.

Una cadena se puede definir con las comillas simples (pero es importante tener en cuenta que no se sustituyen las variables si empleamos comillas simples):
$nombre='juan carlos';

Veremos que en muchos casos se utiliza el concepto de sustitución de variables dentro de un string en PHP por lo que nos acostumbraremos en un principio a utilizar las comillas dobles para definir los string en nuestros programas.

1.3.- Operadores

	Nombre	Resultado
+$a	Identidad	Conversión de $a a int o float según el caso.
-$a	Negación	Opuesto de $a.
$a + $b	Adición	Suma de $a y $b.
$a - $b	Sustracción	Diferencia de $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo	Resto de $a dividido por $b.
$a ** $b	Exponenciación	Resultado de elevar $a a la potencia $ décima. Introducido en PHP 5.6.

El operador de división ("/") devuelve un valor flotante a menos que los dos operandos sean integers (o strings que se conviertan a integers) y los números sean divisibles, en cuyo caso será devuelto un valor integer.
Los operandos del módulo se convierten en integers (por extracción de la parte decimal) antes del procesamiento.
El resultado del operador módulo % tiene el mismo signo que el dividendo — es decir, el resultado de $a % $btendrá el mismo signo que $a. Por ejemplo:

```
<?php

echo (5 % 3)."\n";        // muestra 2
echo (5 % -3)."\n";       // muestra 2
echo (-5 % 3)."\n";       // muestra -2
echo (-5 % -3)."\n";      // muestra -2
```

```
?>
```

Los operadores relacionales disponibles son:
> Mayor
>= Mayor o igual
< Menor
<= Menor o igual
== Igual
!= Distinto

Para ampliar la información relacionada a operadores en PHP consulte el siguiente enlace http://php.net/manual/es/language.operators.php

1.4.- Funciones

La sintaxis para la definición de una función en PHP es:

```
function [nombre de la función]([parámetros])
{
  [algoritmo]
}
```

Implementaremos una función que muestre un título centrado en pantalla, y la llamaremos posteriormente dos veces:

```
<html>n
<head>
<title>Problema</title>
</head>
<body>

<?php
function mostrartitulo($men)
{
  echo "<h1 style=\"text-align:center\">";
  echo $men;
  echo "</h1>";
}

mostrartitulo("Primer título");
echo "<br>";
mostrartitulo("Segundo segundo");

?>
```

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

Para mostrar el titulo centrado utilizamos el elemento h1 de HTML y definimos un estilo centrado para el mismo.
Si vemos la función, notamos que lo más trabajoso es definir todos los elementos HTML para crear el título. Es importante notar que en PHP para introducir las dobles comillas dentro de un string debemos anteceder el carácter ' \'; para introducir el carácter ' \' debemos escribir \\.
Las llamadas a la función las hacemos por medio de su nombre y pasamos el único parámetro que requiere:

```
mostrartitulo("Primer título");
echo "<br>";
mostrartitulo("Segundo segundo");
```

Las funciones nos permiten tener un programa más ordenado y facilitan la reutilización del código.
Más adelante veremos cómo hacer archivos con rutinas comunes a muchas páginas.
Una función puede retornar un dato, supongamos que necesitamos una función que nos retorne el promedio de dos valores, el código sería:

```
<html>
<head>
<title>Problema</title>
</head>
<body>

<?php
function retornarpromedio($valor1,$valor2)
{
  $pro=$valor1/$valor2;
  return $pro;
}

$v1=100;
$v2=50;
$p=retornarpromedio($v1,$v2);
echo $p;
?>

</body>
```

```
</html>
```

Cuando una función retorna un dato debemos emplear la palabra clave return seguida del valor que devuelve.
En la llamada a la función el valor retornado se almacena generalmente en una variable:
$p=retornarpromedio($v1,$v2);

Si queremos que retorne más de un dato debemos emplear parámetros por referencia.
Supongamos que necesitamos ahora que una función nos retorne el cuadrado y cubo de un número:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
function cuadradocubo($valor,&$cuad,&$cub)
{
 $cuad=$valor*$valor;
 $cub=$valor*$valor*$valor;
}
cuadradocubo(2,$c1,$c2);
echo "El cuadrado de 2 es:".$c1."<br>";
echo "El cubo de 2 es:".$c2;
?>
</body>
</html>
```

Es decir, cuando le antecedemos el carácter ampersand al parámetro, es por referencia.
El objetivo es asignarle cierto valor al parámetro y posteriormente el dato quedará almacenado en la variable que le pasamos a la función.

```
function cuadradocubo($valor,&$cuad,&$cub)
{
 $cuad=$valor*$valor;
 $cub=$valor*$valor*$valor;
}
```

El parámetro $cuad se almacena en la variable $c1 y el parámetro $cub se almacena en $c2.
Es la forma más adecuada de modificar variables dentro de una función.

1.5.- Manejo de Archivos

Una actividad fundamental es poder registrar información en el servidor (no como hemos estado haciendo hasta el momento generando sólo una página con los datos cargados)
Para la registración de datos en el servidor disponemos de dos herramientas que se complementan en muchos casos (archivos de texto y bases de datos)
En este apartado veremos cómo crear un archivo de texto y añadir datos al mismo.
Lo presentaremos al tema resolviendo un problema: Implementación de un libro de visitas.
Para resolver este problema plantearemos dos páginas, un formulario para realizar la carga del nombre del visitante y sus comentarios (disponemos un objeto de tipo "text" y otro de tipo "textarea"):

```
<html>
<head>
<title>Problema</title>
</head>
<body> <form action="pagina2.php" method="post">
Ingrese su nombre:
<input type="text" name="nombre">
<br>
Comentarios:
<br>
<textarea name="comentarios" rows="10" cols="40">
</textarea>
<br>
<input type="submit" value="Registrar">
</form>
</body>
</html>
```

Este formulario es similar a los planteados en problemas anteriores, sólo le hemos agregado al control textarea, las propiedades rows y cols que dimensionan el mismo en la pantalla:

```
<textarea name="comentarios" rows="10" cols="40">
</textarea>
```

Veamos ahora la página (pagina2.php) que graba los datos cargados en el formulario en un archivo:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
```

```
 $ar=fopen("datos.txt","a") or
  die("Problemas en la creacion");
 fputs($ar,$_REQUEST['nombre']);
 fputs($ar,"\n");
 fputs($ar,$_REQUEST['comentarios']);
 fputs($ar,"\n");
 fputs($ar,"----------------------------------------------------");
 fputs($ar,"\n");
 fclose($ar);
 echo "Los datos se cargaron correctamente.";
 ?>
</body>
</html>
```

En primer lugar creamos o abrimos el archivo de texto "datos.txt". El segundo parámetro de la función fopen indica la forma de apertura de archivo "a" (lo crea o si ya existe el archivo lo abre para añadir datos al final), "w" (crea el archivo de texto, si existe borra su contenido) y la última forma de apertura del archivo es "r" (abre el archivo para su lectura)
Como en este problema nos interesa que el archivo vaya creciendo con los datos que aportan los visitantes al sitio lo abrimos para añadir, parámetro "a".
La función fopen retorna una referencia al archivo y la almacenamos en una variable llamada $ar.
Si el archivo no se puede abrir, se ejecuta la instrucción que se encuentra luego del operador "or" en nuestro caso llamamos a la función die que finaliza la ejecución del programa PHP mostrando como mensaje el texto que le pasamos a dicha función (por ejemplo si el disco duro del servidor está lleno no se podrá crear el archivo de texto)

```
 $ar=fopen("datos.txt","a") or
   die("Problemas en la creación");
```

Para la grabación de datos utilizamos la función fputs que tiene dos parámetros: la referencia al archivo donde grabamos y el string a grabar.

```
 fputs($ar,$_REQUEST['nombre']);
  fputs($ar,"\n");
```

Para el salto de línea en el archivo de texto, usamos los caracteres \n.De esta forma cuando leamos el archivo de texto lo haremos línea a línea.
Cuando dejamos de trabajar con el archivo llamamos a la función fclose.
Hay que tener muy presente que el archivo se almacena en el servidor y no en la máquina de la persona que está navegando. Es decir, no vaya al explorador de archivos para ver donde se almacenó "datos.txt", tenga en cuenta que está en la máquina donde se ejecutó el script de PHP. Luego veremos como leer el contenido del archivo y mostrarlo en otra página

del sitio (En nuestro caso como utilizamos el equipo como cliente/servidor el archivo datos.txt se crea en la misma carpeta donde se alojan nuestras páginas php)
Para la lectura de un archivo de texto contamos con la función fgets. Además debemos abrir el archivo para lectura.
La apertura para leer:

```
 $ar=fopen("datos.txt","r") or
    die("No se pudo abrir el archivo");
```

Para leer:

```
   $linea=fgets($ar);
```

Veamos como mostrar por pantalla el contenido del archivo "datos.txt" creado en el punto anterior:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
  $ar=fopen("datos.txt","r") or
    die("No se pudo abrir el archivo");
  while (!feof($ar))
  {
    $linea=fgets($ar);
    $lineasalto=nl2br($linea);
    echo $lineasalto;
  }
  fclose($ar);
  ?>
</body>
</html>
```

Lo primero que debemos identificar es la forma de apertura del archivo:

```
 $ar=fopen("datos.txt","r") or
  die("No se pudo abrir el archivo");
```

El segundo parámetro de fopen es "r" es decir read (apertura para lectura), si el archivo no existe por ejemplo se ejecuta la función die que finaliza el programa mostrando el string correspondiente.
La función feof retorna true si se ha llegado al final del archivo en caso contrario retorna false. Para que se impriman todas las líneas del archivo se plantea una estructura repetitiva

que se ejecuta mientras no se llegue al final de archivo (el operador lógico not en PHP es el caracter !):

```
while (!feof($ar))
```

Dentro de la estructura repetitiva leemos una línea completa del archivo de texto con la función fgets:

```
$linea=fgets($ar);
```

La variable $linea contiene una línea completa del archivo de texto, inclusive el salto de línea (\n)

Como el navegador no hace un salto de línea con este carácter, debemos convertir dicho caracter al elemento
 propia de HTML. La función que realiza esta actividad se llama nl2br (new line to br)

El resultado se almacena en una nueva variable que es la que realmente imprimimos:

```
$lineasalto=nl2br($linea);
echo $lineasalto;
```

1.6.- Cookies y Sesiones

El protocolo HTTP es desconectado. Esto significa que cada vez que solicitamos una página a un servidor representa una conexión distinta.

Una cookie es una pequeña cantidad de datos almacenada por el navegador del usuario cuando solicita una página a un servidor. El que envía que se genere la cookie es el servidor.

Una cookie consta de un nombre, un valor, una fecha de expiración y un servidor. Una cookie está limitada a 4KB.

Luego que una cookie es creada sólo el sitio que la creó puede leerla. Luego de creada una cookie, cada vez que el navegador del usuario visita el sitio, se envía dicha cookie. Otra cosa importante que hay que tener en cuenta es que el usuario del browser puede configurar el mismo para no permitir la creación de cookies, lo que significa que el uso de cookies debe hacerse con moderación y cuando la situación lo requiera. De todos modos, el 95% de los navegadores están configurados para permitir la creación de cookies.

Para la creación de una cookie desde PHP debemos llamar a la función setcookie.

Los parámetros de esta función son:

setcookie(<nombre de la cookie>, <valor de la cookie>, <fecha de expiración>, <carpeta del servidor>)

Con un problema sencillo entenderemos el uso de esta función. Supongamos que queremos que los usuarios que entran a nuestro sitio puedan configurar con qué color de fondo de página quiere que aparezca cada vez que ingresa al sitio. Al color seleccionado por el visitante lo almacenaremos en una cookie. En caso que no exista el color, por defecto es blanco.

La primera página mostrará un formulario con tres controles de tipo radio para la selección del color. También esta página verificará si existe la cookie creada, en caso afirmativo fijará el fondo de la página con el valor de la cookie. Tengamos en cuenta que la primera vez que ejecutemos este programa la página es de color blanco, luego variará según el color seleccionado en el formulario.

El código de la primera página es:

```
<html>
<head>
<title>Problema</title>
</head>
<body
<?php if (isset($_COOKIE['color'])) echo " style=\"background:$_COOKIE[color]\"" ?>
>
<form action="pagina2.php" method="post">
Seleccione de que color desea que sea la página de ahora en más:<br>
<input type="radio" value="rojo" name="radio">Rojo<br>
<input type="radio" value="verde" name="radio">Verde<br>
<input type="radio" value="azul" name="radio">Azul<br>
<input type="submit" value="Crear cookie">
</form>
</body>
</html>
```

El formulario no varía en nada respecto a otros vistos. Lo más importante es el bloque PHP que verifica si ya existe la cookie en el navegador del cliente. Es importante entender que la primera vez que ejecutemos esta página la cookie no existe, por lo que el if se verifica falso:

```
<body
<?php if (isset($_COOKIE['color'])) echo " style=\"background:$_COOKIE[color]\"" ?>
>
```

El vector asociativo $_COOKIE almacena todas las cookies creadas por el visitante. Si es la primera vez que peticionamos esta página, el vector $_COOKIE no tendrá elementos.

Es decir que la marca body no tiene inicializada la propiedad style.

La segunda página es la que crea la cookie propiamente dicha:

```
<?php
if ($_REQUEST['radio']=="rojo")
 setcookie("color","#ff0000",time()+60*60*24*365,"/");
elseif ($_REQUEST['radio']=="verde")
 setcookie("color","#00ff00",time()+60*60*24*365,"/");
elseif ($_REQUEST['radio']=="azul")
 setcookie("color","#0000ff",time()+60*60*24*365,"/");
```

```
?>
<html>
<head>
<title>Problema</title>
</head>
<body>
Se creó la cookie.
<br>
<a href="pagina1.php">Ir a la otra página</a>
</body>
</html>
```

La llamada a la función setcookie debe hacerse antes de imprimir cualquier etiqueta HTML, de lo contrario no funcionará.
Como podemos observar, la creación de la cookie se hace llamando a la función setcookie:

```
<?php
if ($_REQUEST['radio']=="rojo")
  setcookie("color","#ff0000",time()+60*60*24*365,"/");
elseif ($_REQUEST['radio']=="verde")
  setcookie("color","#00ff00",time()+60*60*24*365,"/");
elseif ($_REQUEST['radio']=="azul")
  setcookie("color","#0000ff",time()+60*60*24*365,"/");
?>
```

El nombre de la cookie se llama "color" y el valor que almacenamos depende de qué control de tipo radio esté seleccionado en la página anterior. La fecha de expiración de la cookie la calculamos fácilmente llamando a la función time() que nos retorna la fecha actual en segundos y le sumamos el producto 60*60*24*365 (60 segundos * 60 minutos * 24 horas * 365 días) es decir que la cookie existirá en la máquina del visitante hasta el año próximo.
Cuando indicamos como directorio la sintaxis "/" significa que la cookie se crea a nivel del sitio y con cualquier petición a dicho sitio, el navegador enviará la cookie al servidor.
Por último dispusimos en esta página un hipervínculo a la página anterior, para ver que, de ahora en más, cada vez que ejecutemos la pagina1.php, el color de fondo de la misma dependerá del valor de la cookie registrada.

EVALUACIÓN

Ejercicios
Generar un valor aleatorio entre 1 y 3. Luego imprimir en castellano el número (Ej. si se genera el 3 luego mostrar en la página el string "tres").

Definir una variable de cada tipo: integer, double, string y boolean. Luego imprimirlas en la página, una por línea.
Confeccionar un programa en PHP que permita hacer el pedido de pizzas vía internet. El formulario debe ser:
Nombre:[.....]
Direccion:[.....]
Jamon y queso:[x]
Cantidad[...]

Napolitana:[x]
Cantidad[...]

Mozzarella;[x]
Cantidad[...]

[Confirmar]

Para el ingreso del nombre, dirección y cantidad de pizzas de cada tipo disponer objetos de la clase "text".
Disponer tres objetos de tipo "checkbox" para seleccionar los tipos de pizzas.
Por último disponer un botón para el envío de datos: "submit".
Grabar en un archivo de texto llamado "pedidos.txt" cada pedido, separados por una línea de puntos entre cada pedido

UNIDAD 2 PHP y MySQL

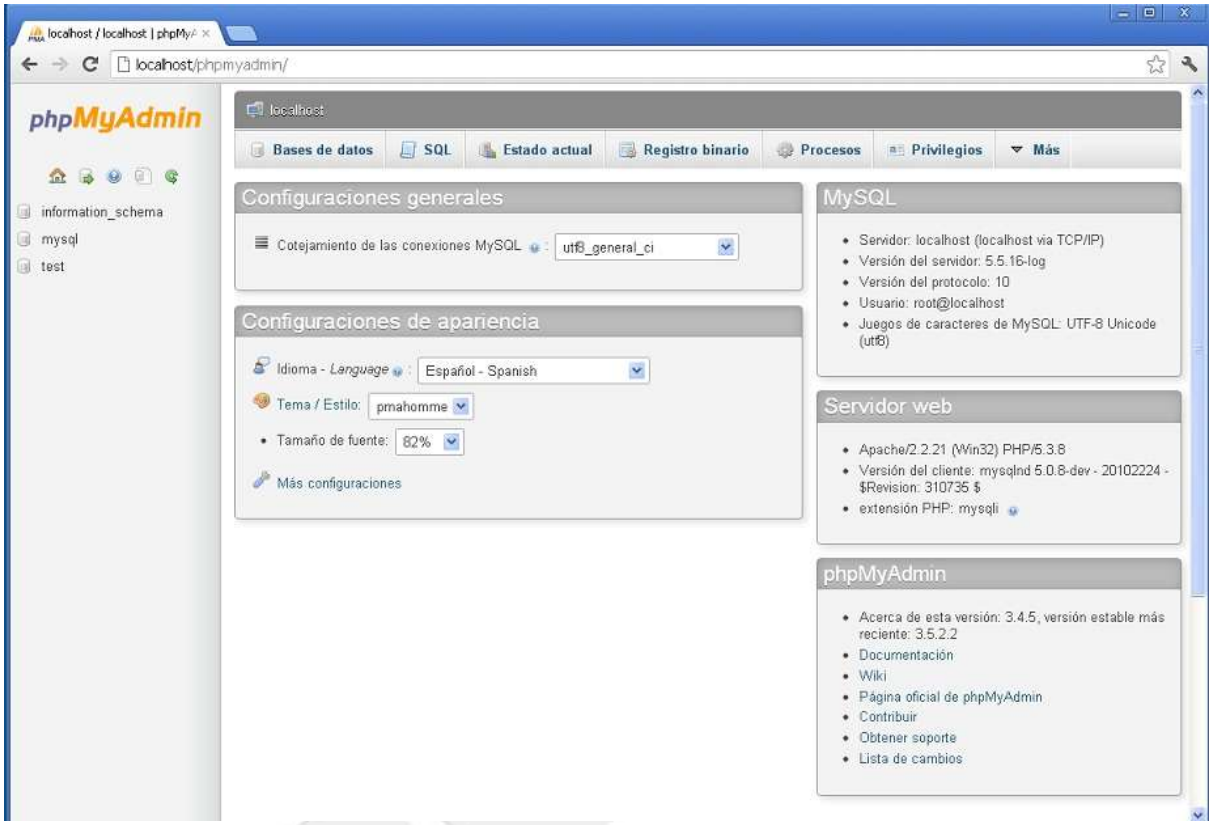
2.1.- Primeros Pasos en MySQL

Uno de los empleos principales de PHP es el acceso a una base de datos en el servidor. Las operaciones básicas se hacen empleando como lenguaje el SQL.
PHP implementa distintas funciones según la base de datos a emplear. Existen funciones actualmente para acceder a los siguientes servidores de bases de datos:
- MySQL
- Microsoft SQL Server
- Oracle
- PostgreSQL
- SysBase
- FrontBase
- Informix
- InterBase

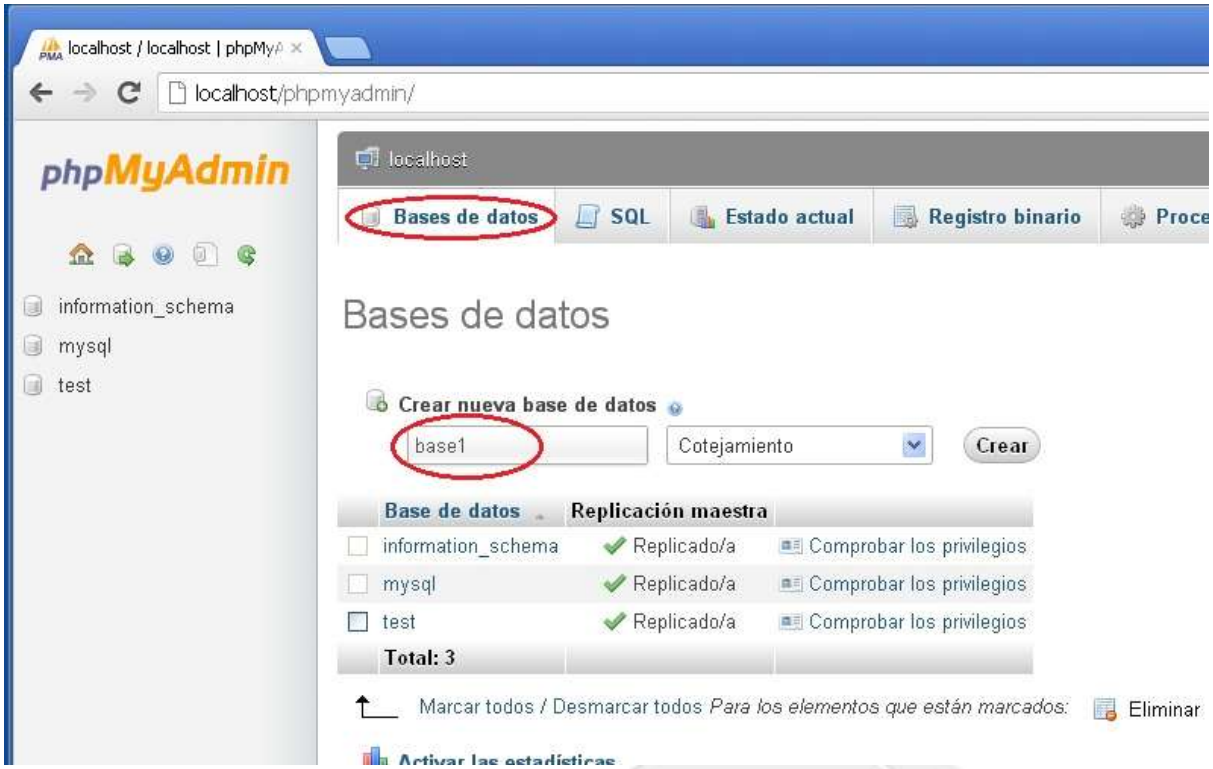
- Ingres
- mSQL
- dBase
- SQLlite

El más empleado en la actualidad en la web es el gestor de base de datos MySQL.

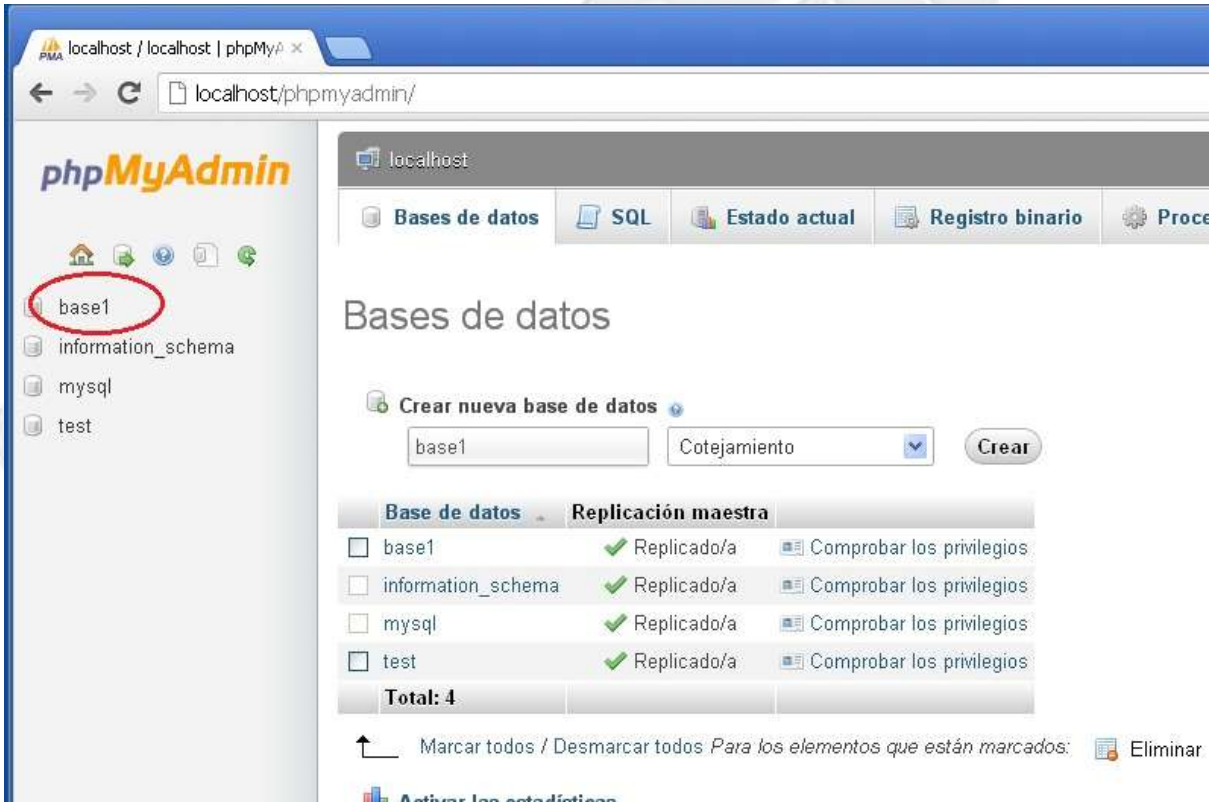
PHPMyAdmin es un programa que se ejecuta en la web:



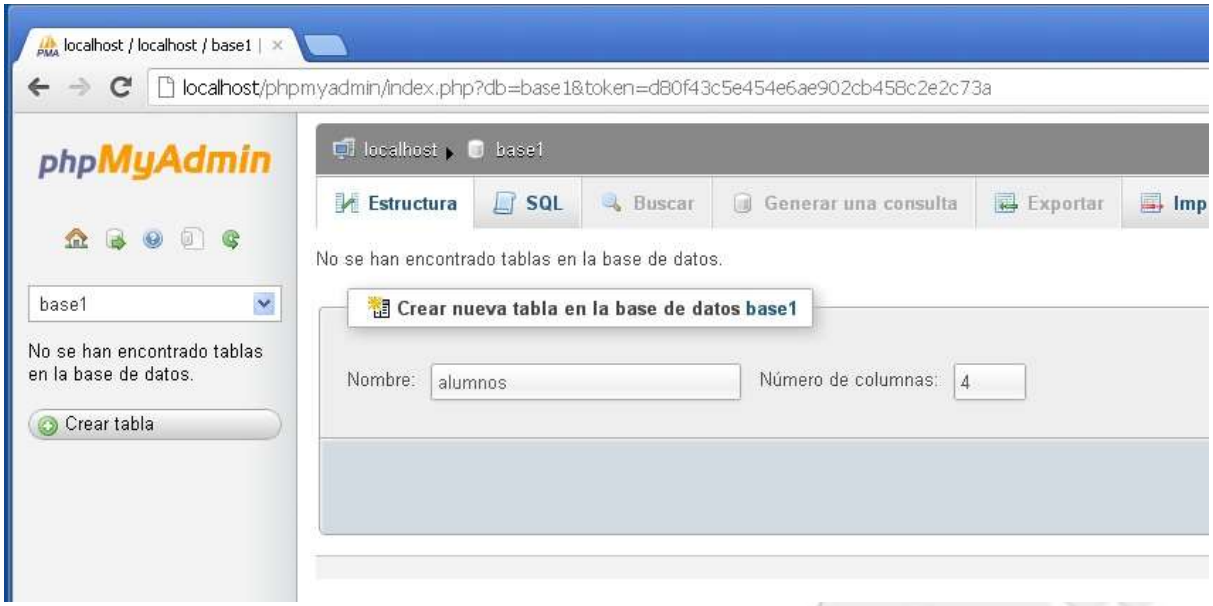
Para crear una base de datos procedemos a seleccionar la pestaña "Base de datos" e ingresamos como nombre "base1" y presionamos el botón crear:



Luego de crear la base de datos podemos ver que aparece en el lado izquierdo:



Seleccionamos el nombre de la base de datos "base1" y se actualiza la interfaz de pantalla para que ingresemos el nombre de una tabla y la cantidad de campos que tendrá (crearemos una tabla llamada alumnos con 4 campos):



La estructura de la tabla es:

codigo int auto_increment primary key

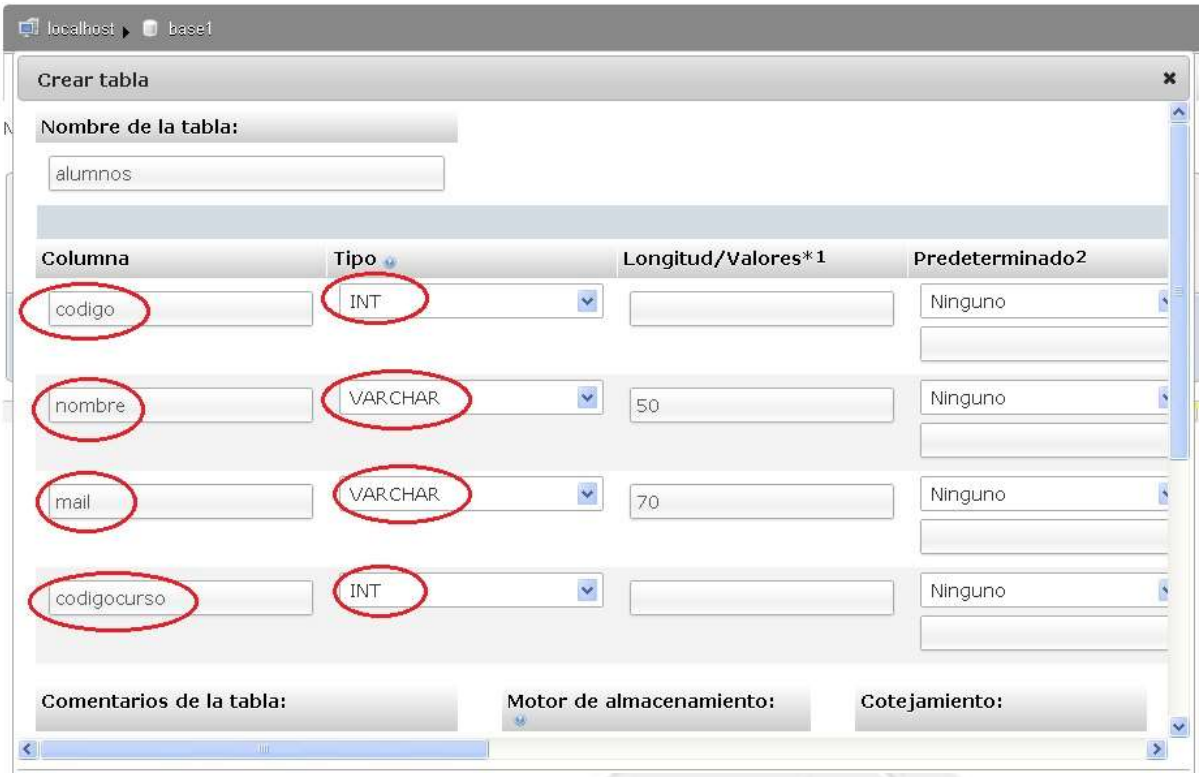
nombre varchar(50)

mail varchar(70)

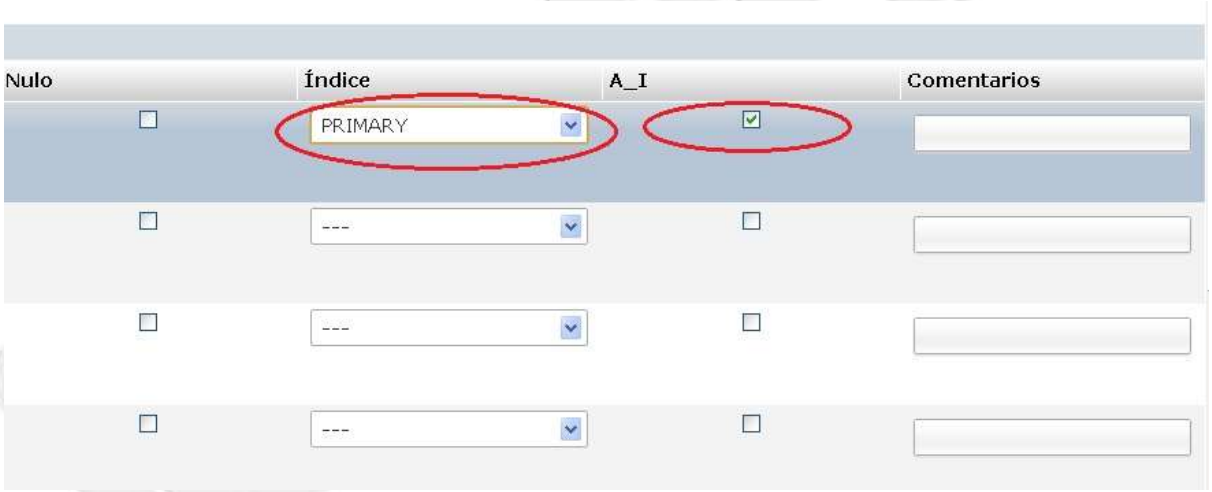
codigocurso int

2.2.- Creación de una Base de Datos

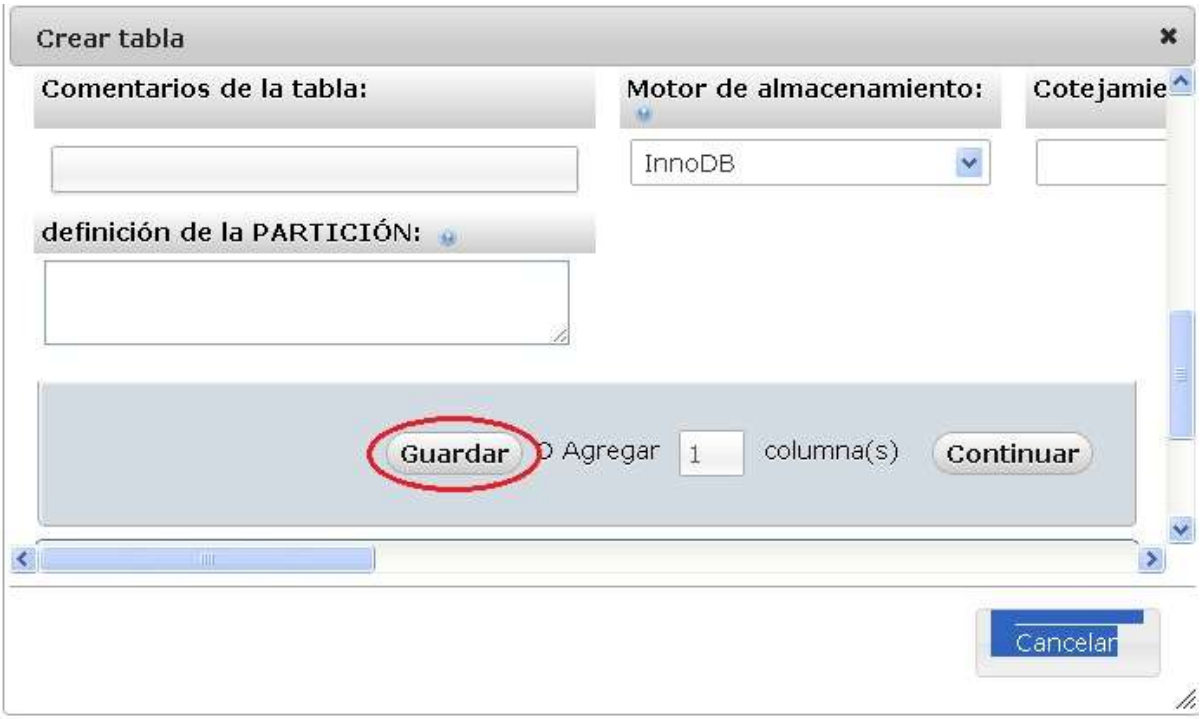
En el PHPMyAdmin ingresamos:



Es importante también hacer notar que en el campo código debemos marcar en Índice el valor "primary" y tildar la columna A_I:



Por último presionamos el botón guardar y ya tenemos la tabla "alumnos" creada en la base de datos "base1":



La tabla almacenará datos de alumnos que desarrollarán cursos de programación en PHP, ASP y JSP.

El código del alumno es de tipo numérico (int) y al indicar que es auto_increment se generará automáticamente por el gestor de base de datos.

Los campos nombre y mail son de tipo varchar (podemos almacenar cualquier caracter) y por último el campo codigocurso representa el curso a tomar por el alumno (1=PHP, 2=ASP y 3=JSP)

El campo clave de esta tabla es el código de alumno (es decir no podemos tener dos alumnos con el mismo código, no así el nombre del alumno que puede eventualmente repetirse)

2.3 Insertar datos

Luego de crear una base de datos y sus tablas (Vamos a trabajar con la base de datos ya creada: base1, que contiene la tabla alumnos), veremos como agregar registros.

Para añadir datos en la tabla empleamos el comando SQL llamado insert.

Necesitamos dos páginas para este proceso, una será el formulario de carga de datos y la siguiente será la que efectúe la inserción en la tabla.

Formulario de carga de datos:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<h1>Alta de Alumnos</h1>
```

```
<form action="pagina2.php" method="post">
Ingrese nombre:
<input type="text" name="nombre"><br>
Ingrese mail:
<input type="text" name="mail"><br>
Seleccione el curso:
<select name="codigocurso">
<option value="1">PHP</option>
<option value="2">ASP</option>
<option value="3">JSP</option>
</select>
<br>
<input type="submit" value="Registrar">
</form>
</body>
</html>
```

El formulario es bastante similar a los que venimos desarrollando en puntos anteriores, tal vez lo distinto es cómo emplearemos el control "select" del curso a desarrollar:

```
<select name="codigocurso">
<option value="1">PHP</option>
<option value="2">ASP</option>
<option value="3">JSP</option>
</select>
```

Cada opción tiene su respectivo valor (en este caso los números 1,2 y 3) y los textos a mostrar PHP, ASP y JSP. El dato que se envía a la otra página es el código de curso (esto debido a que definimos la propiedad value).

Ahora veremos cómo realizar la registración de los datos cargados en el formulario, en la tabla alumnos de la base de datos base1:

```
<html>
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
$conexion=mysqli_connect("localhost","root","","base1") or
   die("Problemas con la conexión");

mysqli_query($conexion,"insert into alumnos(nombre,mail,codigocurso) values
```

```
                ('$_REQUEST[nombre]','$_REQUEST[mail]',$_REQUEST[codigocurso])")
  or die("Problemas en el select".mysqli_error($conexion));

mysqli_close($conexion);
echo "El alumno fue dado de alta.";
?>
</body>
</html>
```

Veamos los pasos para efectuar el alta en la tabla alumnos:

```
$conexion=mysqli_connect("localhost","root","","base1")                    or
   die("Problemas con la conexión");
```

La función mysqli_connect se conecta a una base de datos de tipo MySql, el primer parámetro es la dirección donde se encuentra el gestor de base de datos (en este caso en el mismo servidor por lo que indicamos esto con "localhost), el segundo parámetro es el nombre de usuario de la base de datos ("root" en nuestro caso, que es el usuario por defecto que crea MySql para el administrador), seguidamente indicamos la clave del usuario root (por defecto al instalar el Wamp se crea con clave vacía) y por último indicamos el nombre de la base de datos a conectarnos (en nuestro ejemplo ya creamos la base de datos llamada: base1 que tiene la tabla alumnos)

En caso de haber algún error en la llamada a la función la misma retorna false por lo que se ejecuta la instrucción seguida del operador or, en nuestro caso llamamos a la función die que detiene la ejecución del programa y muestra el mensaje por pantalla.

El paso más importante es la codificación del comando SQL insert(debemos llamar a la función mysqli_query pasando como primer parámetro la referencia a la conexion y el segundo parámetro es un string con el comando insert):

```
mysqli_query($conexion,"insert into alumnos(nombre,mail,codigocurso) values
                ('$_REQUEST[nombre]','$_REQUEST[mail]',$_REQUEST[codigocurso])")
  or die("Problemas en el select".mysqli_error($conexion));
```

La sintaxis del comando insert es bastante sencilla, indicamos el nombre de la tabla y los campos de la tabla a cargar. Luego debemos indicar en el mismo orden los valores a cargar en cada campo (dichos valores los rescatamos del formulario anterior).

Los campos de tipo varchar SQL requiere que encerremos entre comillas simples, esto sucede para el nombre y el mail; en cambio, para el codigocurso esto no debe ser así.

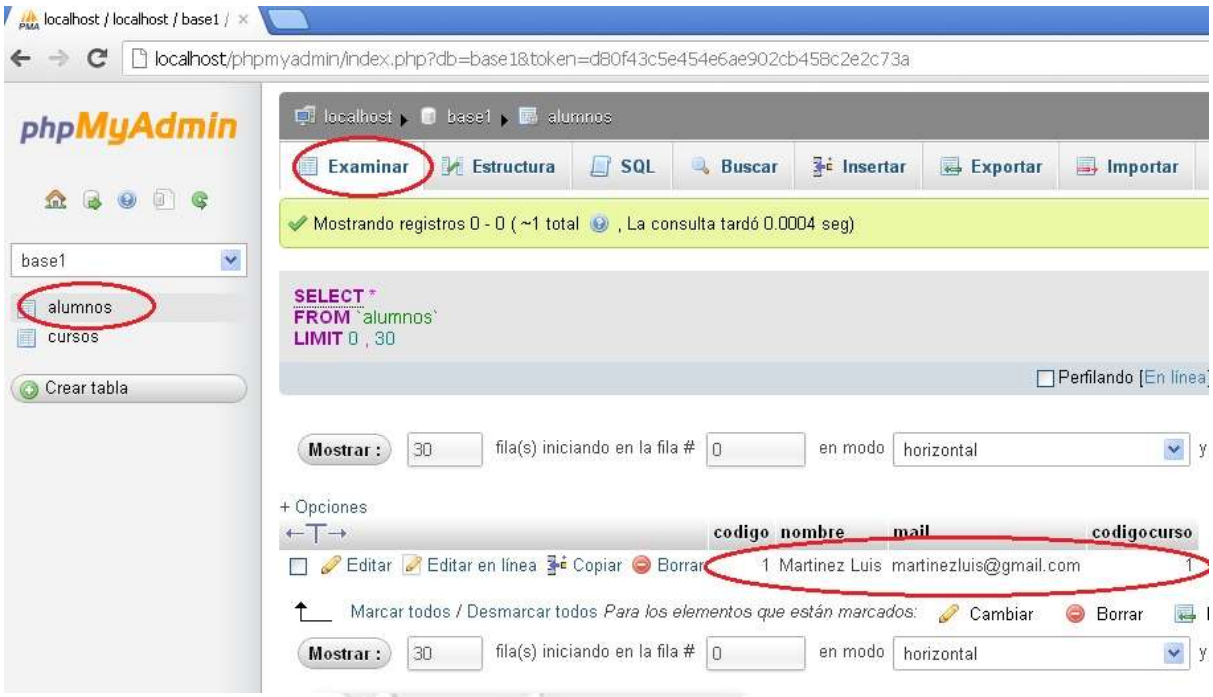
Otra cosa a tener en cuenta es que los subíndices de los vectores asociativos no deben ir entre simples comillas ya que se encuentran dentro de un string, sino se producirá un error.

En caso que MySql detecte un error, retorna false esta función, por lo que se ejecuta la instrucción posterior al or, es decir la función die que mostrará el error generado por MySql llamando a la función mysqli_error().

Por último cerramos la conexión con la base de datos y mostramos un mensaje indicando que la carga se efectuó en forma correcta.

Tener en cuenta que el campo código se generó en forma automática.

Si queremos controlar que el insert se efectuó en forma correcta podemos ingresar al PHPMyAdmin y seleccionar la tabla "alumnos", y en la pestaña "examinar" podremos ver los datos ingresados desde la página que creamos:



2.4.- Consultar datos de una Tabla

El proceso de consulta de datos de una tabla es similar al del listado, la diferencia es que se muestra sólo aquel o aquellos que cumplen la condición por la que buscamos.

Haremos un programa que nos permita consultar los datos de un alumno ingresando su mail para su búsqueda. Tengamos en cuenta que no puede haber dos alumnos con el mismo mail, por lo que la consulta nos puede arrojar uno o ningún registro.

Debemos codificar un formulario para el ingreso del mail a consultar:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<form action="pagina2.php" method="post">
Ingrese el mail del alumno a consultar:
<input type="text" name="mail">
<br>
<input type="submit" value="buscar">
```

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

Por otro lado tenemos el archivo "pagina2.php" que se encarga de buscar el mail ingresado en el formulario:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
$conexion=mysqli_connect("localhost","root","","base1") or
   die("Problemas con la conexión");

$registros=mysqli_query($conexion,"select codigo,nombre, codigocurso
                from alumnos where mail='$_REQUEST[mail]'") or
  die("Problemas en el select:".mysqli_error($conexion));

if ($reg=mysqli_fetch_array($registros))
{
  echo "Nombre:".$reg['nombre']."<br>";
  echo "Curso:";
  switch ($reg['codigocurso']) {
    case 1:echo "PHP";
          break;
    case 2:echo "ASP";
          break;
    case 3:echo "JSP";
          break;
  }
}
else
{
  echo "No existe un alumno con ese mail.";
}
mysqli_close($conexion);
?>
</body>
</html>
```

Lo más importante está en el comando select:

```
$registros=mysqli_query($conexion,"select codigo,nombre, codigocurso
                from alumnos where mail='$_REQUEST[mail]'") or
  die("Problemas en el select:".mysqli_error($conexion));
```

Acá es donde con la cláusula where seleccionamos sólo el registro que cumple con la condición que el mail sea igual al que ingresamos.
Como sólo puede haber un registro que cumpla la condición, llamamos a la función mysqli_fetch_array en un if y no una estructura repetitiva como el listado:

```
if ($reg=mysqli_fetch_array($registros))
```

En caso de retornar un vector asociativo la condición del if se verifica como verdadera y pasa a mostrar los datos, en caso de retornar false se ejecuta el else.

2.5.- Actualizar datos de una Tabla

De las actividades con una tabla esta es la más larga. Vamos a resolverlo implementando tres páginas, la primera un formulario de consulta del mail de un alumno, la segunda otro formulario que nos permita cargar su mail modificado y la última registrará el cambio en la tabla.
El formulario de consulta del mail del alumno es similar a problemas anteriores:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<form action="pagina2.php" method="post">
Ingrese el mail del alumno:
<input type="text" name="mail"><br>
<input type="submit" value="buscar">
</form>
</body>
</html>
```

La segunda página es la más interesante y con conceptos nuevos:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
```

```
<?php

$conexion=mysqli_connect("localhost","root","","base1") or
   die("Problemas con la conexión");

$registros=mysqli_query($conexion,"select * from alumnos
                where mail='$_REQUEST[mail]'") or
  die("Problemas en el select:".mysqli_error($conexion));
if ($reg=mysqli_fetch_array($registros))
{
?>

<form action="pagina3.php" method="post">
Ingrese nuevo mail:
<input type="text" name="mailnuevo" value="<?php echo $reg['mail'] ?>">
<br>
<input type="hidden" name="mailviejo" value="<?php echo $reg['mail'] ?>">
<input type="submit" value="Modificar">
</form>

<?php
}
else
  echo "No existe alumno con dicho mail";
?>
</body>
</html>
```

Lo primero que podemos observar es que si el if se verifica verdadero se ejecuta un bloque que contiene código HTML:

```
if ($reg=mysqli_fetch_array($registros))
{
?>
  <form action="pagina3.php" method="post">
  Ingrese nuevo mail:
  <input type="text" name="mailnuevo" value="<?php echo $reg['mail'] ?>">
  <br>
  <input type="hidden" name="mailviejo" value="<?php echo $reg['mail'] ?>">
  <input type="submit" value="Modificar">
  </form>
<?php
```

```
}
```

Es decir que podemos disponer bloques de PHP dispersos dentro de la página.
Otro concepto importante es como enviar el mail del primer formulario a la tercera página, esto se logra con los controles de tipo "hidden", este tipo de control no se visualiza en el formulario pero se envía al presionar el botón submit.
Si queremos que el control text se cargue con el mail ingresado en el formulario anterior debemos cargar la propiedad value con dicho valor:

```
 <input type="text" name="mailnuevo" value="<?php echo $reg['mail'] ?>">
```

Por último la pagina3.php es la que efectúa la modificación de la tabla propiamente dicha. Con el mail ingresado en la pagina1.php, el mail modificado en la pagina2.php se efectúa el update.

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
$conexion=mysqli_connect("localhost","root","","base1") or
   die("Problemas con la conexión");

mysqli_query($conexion, "update alumnos
                    set mail='$_REQUEST[mailnuevo]'
                  where mail='$_REQUEST[mailviejo]'") or
  die("Problemas en el select:".mysqli_error($conexion));
  echo "El mail fue modificado con exito";
?>
</body>
</html>
```

Tengamos en cuenta que el segundo formulario nos envía dos datos: $_REQUEST[mailnuevo] y $_REQUEST[mailviejo].

2.6.- Borrar datos de una Tabla

El objetivo de este punto es el borrado de un registro de una tabla. Para ello, implementaremos un algoritmo que solicite ingresar el mail de un alumno y posteriormente efectúe su borrado.
Para eliminar filas en una tabla debemos utilizar el comando SQL delete.

La primera página es idéntica a la consulta, ya que debemos implementar un formulario que solicite la carga del mail del alumno:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<form action="pagina2.php" method="post">
Ingrese el mail del alumno a borrar:
<input type="text" name="mail">
<br>
<input type="submit" value="buscar">
</form>
</body>
</html>
```

Por otro lado tenemos el archivo "pagina2.php" que se encarga de buscar el mail ingresado en el formulario y en caso que exista se procede a borrarlo:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
$conexion=mysqli_connect("localhost","root","","base1") or
   die("Problemas con la conexión");

$registros=mysqli_query($conexion, "select codigo from alumnos
                where mail='$_REQUEST[mail]'") or
  die("Problemas en el select:".mysqli_error($conexion));
if ($reg=mysqli_fetch_array($registros))
{
  mysqli_query($conexion,"delete from alumnos where mail='$_REQUEST[mail]'") or
    die("Problemas en el select:".mysqli_error($conexion));
  echo "Se efectuó el borrado del alumno con dicho mail.";

}
else
{
  echo "No existe un alumno con ese mail.";
```

```
}
mysqli_close($conexion);
?>
</body>
</html>
```

En esta segunda página efectuamos dos llamadas a la función mysqli_query, una para consultar si existe el mail ingresado y otra para efectuar el borrado del registro respectivo. Si no existe el mail ingresado mostramos un mensaje de tal situación.

2.7.- Borrar una Tabla

Vimos en el concepto anterior que podemos borrar una o más filas de una tabla indicando en el where del comando SQL delete la condición que debe cumplir la fila para ser borrada.
Para borrar todos los registros de una tabla debemos llamar al comando delete de SQL sin disponer la cláusula where.
Es importante tener en cuenta que la ausencia de la cláusula where en el comando delete eliminará todas las filas de la tabla.

```
<html>
<head>
<title>Problema</title>
</head>
<body>
<?php
$conexion=mysqli_connect("localhost","root","","base1") or
   die("Problemas con la conexión");

mysqli_query($conexion,"delete from alumnos") or
  die("Problemas en el select:".mysqli_error($conexion));
echo "Se efectuó el borrado de todos los alumnos.";
mysqli_close($conexion);
?>
</body>
</html>
```

EVALUACIÓN

1) Efectuar la modificación del nombre del curso de la tabla "cursos". Para la búsqueda ingresar el código de curso.
2) Efectuar el borrado de todos los registros de la tabla cursos.

UNIDAD 3. Programación Orientada a Objetos

3.1.- Manejo de Clases

Clases

Esta extensión nos suministra dos clases fundamentales:

mysqli

mysqli_result

Para conectarnos con el motor de base de datos y seleccionar una base de datos debemos crear un objeto de la clase mysqli:

```
$mysql=new mysqli("localhost","root","","base1");
```

Para crear un objeto debemos utilizar el operador new e inmediatamente disponer el nombre de la clase (en este caso la clase se llama mysqli) y entre paréntesis los parámetros del constructor. El contructor recibe los mismos parámetros que habíamos visto con la interfaz procedimental, es decir el nombre del servidor, el usuario, la clave y el nombre de la base de datos.

La variable $mysql almacena la referencia del objeto que se acaba de crear de la clase mysqli.

Para saber si la conexión y selección de la base de datos se efectuó en forma correcta debemos acceder a la propiedad connect_error de la clase mysqli, dispondremos una sintaxis similar a esta:

```
if ($mysql->connect_error)
  die('Problemas con la conexion a la base de datos');
```

$mysql es un objeto de la clase mysqli y mediante este objeto accedemos a la propiedad connect_error, si el if se verifica falso es que la propiedad connect_error almacena un NULL ya que no hay un error en la conexión al servidor y base de datos indicado cuando creamos el objeto.

Para ejecutar una consulta la clase mysqli tiene un método llamado query, la sintaxis para llamar a este método tendrá una estructura similar a:

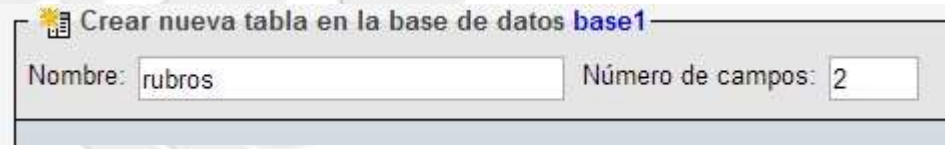
```
$mysql->query("insert into rubros(descripcion) values ('$_REQUEST[descripcion]')") or
  die($mysql->error);
```

Al método query le pasamos como referencia un string con el comando SQL que queremos que se ejecute, si hay algún error en el comando SQL el método query retorna un false y se ejecuta el comando seguido al operador or.
Con la función die detenemos la ejecución del programa y accedemos a la propiedad error que almacena el error generado al tratar de ejecutar el comando SQL.
Finalmente llamamos al método close para cerrar la conexión:
$mysql->close();

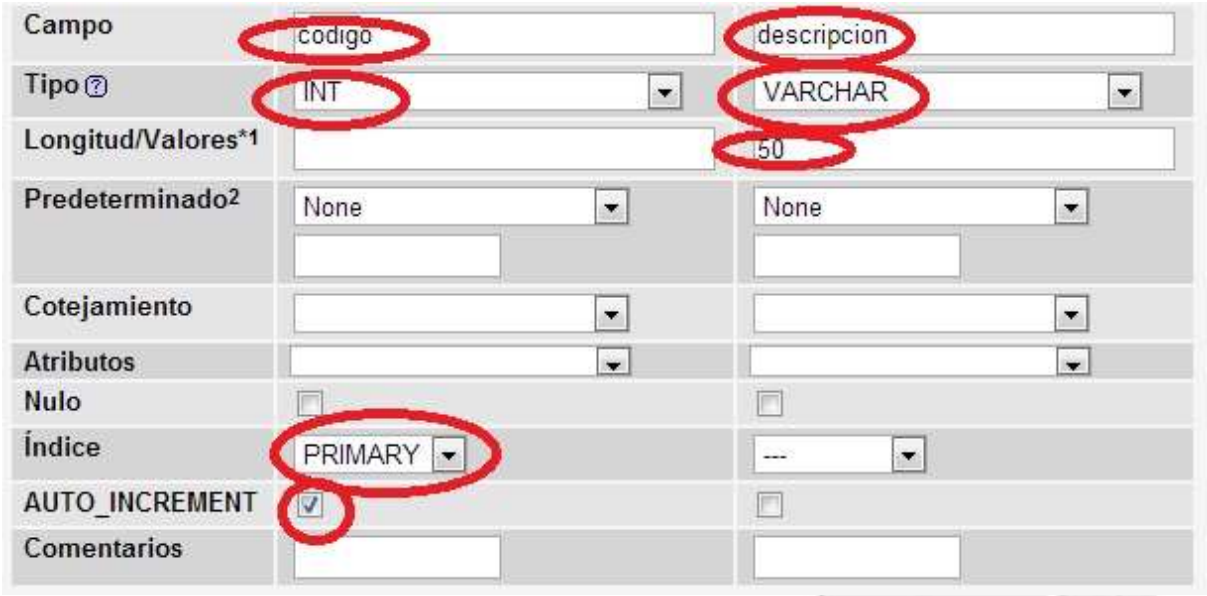
3.2 Uso de PHP MyAdmind

Ejemplo
Problema
Implementaremos una serie de páginas para efectuar el ABM (Altas, Bajas, y Modificaciones) y listados de una serie de tablas empleando la extensión mysqli orientada a objetos.
Crearemos una base de datos base1 (si ya la tenemos de conceptos anteriores procederemos a agregar las tablas a dicha base de datos)

Crearemos dos tablas llamadas "rubros" y "articulos", en la primera almacenaremos un listado de rubros de un supermercado y en la tabla "articulos" almacenaremos la descripción, precio y código de rubro que pertenece el artículo.
Desde el PhpMyAdmin procedemos a crear las dos tablas, primero indicamos el nombre de la tabla:

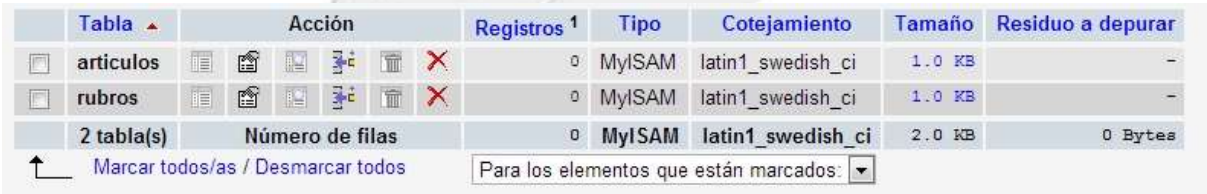


y seguidamente indicamos la estructura de la tabla rubros:



y la tabla articulos tiene la siguiente estructura:

Ya tenemos las dos tablas creadas dentro de nuestra base de datos "base1":



Para recuperar datos de tablas disponemos de una clase llamada mysqli_result.
La clase mysqli_result tiene un método fetch_array que nos permiten acceder a los campos rescatados con el comando SQL Select.
Implementaremos el listado completo de la tabla rubros.

```
<!doctype html>
<html>
<head>
 <title>Listado</title>
 <style>
 .tablalistado                                                    {
  border-collapse: collapse;
        box-shadow: 0px 0px 8px #000;
```

```
        margin:20px;
  }
  .tablalistado th{
    border: 1px solid #000;
    padding: 5px;
    background-color:#ffd040;
  }

  .tablalistado td{
    border: 1px solid #000;
    padding: 5px;
    background-color:#ffdd73;
  }
  </style>
</head>
<body>

  <?php
    $mysql=new mysqli("localhost","root","","base1");
        if ($mysql->connect_error)
          die("Problemas con la conexión a la base de datos");

    $registros=$mysql->query("select codigo,descripcion from rubros") or
          die($mysql->error);

    echo '<table class="tablalistado">';
    echo '<tr><th>Código</th><th>Descripción</th></tr>';
    while ($reg=$registros->fetch_array())
    {
      echo '<tr>';
      echo '<td>';
      echo $reg['codigo'];
      echo '</td>';
      echo '<td>';

      echo $reg['descripcion'];
      echo '</td>';
      echo '</tr>';
    }
    echo '<table>';
```

```
   $mysql->close();

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

Similar a la página anterior creamos un objeto de la clase mysqli y verificamos si no hubo error:

```
  $mysql=new mysqli("localhost","root","","base1");
   if ($mysql->connect_error)
     die('Problemas con la conexion a la base de datos');
```

Ahora llamamos al método query pasando un string con el comando SQL select. El método query retorna un objeto de la clase mysqli_result:

```
  $registros=$mysql->query("select codigo,descripcion from rubros") or
           die($mysql->error);
```

Si el método query de la clase mysqli retorna un false significa que hubo un error en el comando SQL select y pasa a ejecutar la instrucción seguida al operador or, en este caso con el die detenemos la ejecución del programa y accedemos a la propiedad error del objeto $mysql para mostrarlo.

El objeto de la clase mysqli_result tiene un método llamado fetch_array que retorna de a uno cada una de los registros recuperados con el comando SQL select:

```
  while ($reg=$registros->fetch_array())
   {
     echo '<tr>';
     echo '<td>';
     echo $reg['codigo'];
     echo '</td>';
     echo '<td>';
     echo $reg['descripcion'];
     echo '</td>';
     echo '</tr>';
   }
```

En pantalla obtenemos como resultado:

Código	Descripción
1	frutas
2	verduras
3	carnes

EVALUACIÓN

Actividad final

Crearemos tres páginas. La primera donde el operador ingresa el código del artículo que quiere modificar, la segunda página mostraremos un formulario HTML con los datos precargados y finalmente la tercer página tendrá como objetivo ejecutar el comando SQL update.

REFERENCIAS

http://php.net/manual/es/language.operators.php
https://www.tutorialesprogramacionya.com/phpya/index.php?inicio=0