UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

Diplomatura: Proffessional backend developer

Curso: Programador Web avanzado

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 1: Introducción y nivelación

Unidad 1: Nivelación PHP y MySQL

Presentación:

El desarrollo de aplicaciones web backend implica conocimientos básicos necesarios sobre PHP y MySql. Parte de esos conocimientos son el manejo de variables, funciones, vectores, parámetros, variables globales entre otros. Por el lado de MySql nos encontramos con conceptos como tablas, claves primarias y foráneas, relación entre tablas. Es imprescindible comprender estos conceptos para lo que resta del curso, ya que la utilización de los mismos será necesaria para los nuevos temas.

Objetivos:

Que los participantes:

- Obtengan conocimientos sobre los lenguajes básicos para desarrollo web.
- Incorporen conocimientos sobre conceptos básicos de la programación web.
- Sepan aplicar y utilizar variables y funciones en PHP.
- Aprendan a crear y modificar una tabla.

Bloques temáticos:

- Introducción y nivelación de PHP.
- Uso de variables y constantes en PHP.
- Condicionales.
- Ciclos.
- Vectores.
- Formularios.
- Session.
- Base de datos.
- MySql.
- Operaciones básicas sobre registros en MySql.
- Tipos de datos en MySql.
- Operadores.
- Funciones.
- Joins.
- PHPMYADMIN.

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

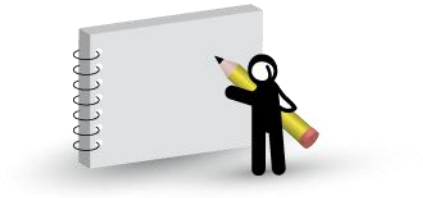
- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

** El MEC es el modelo de E-learning colaborativo de nuestro Centro.*



Tomen nota:

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

- **Introducción y nivelación de PHP**

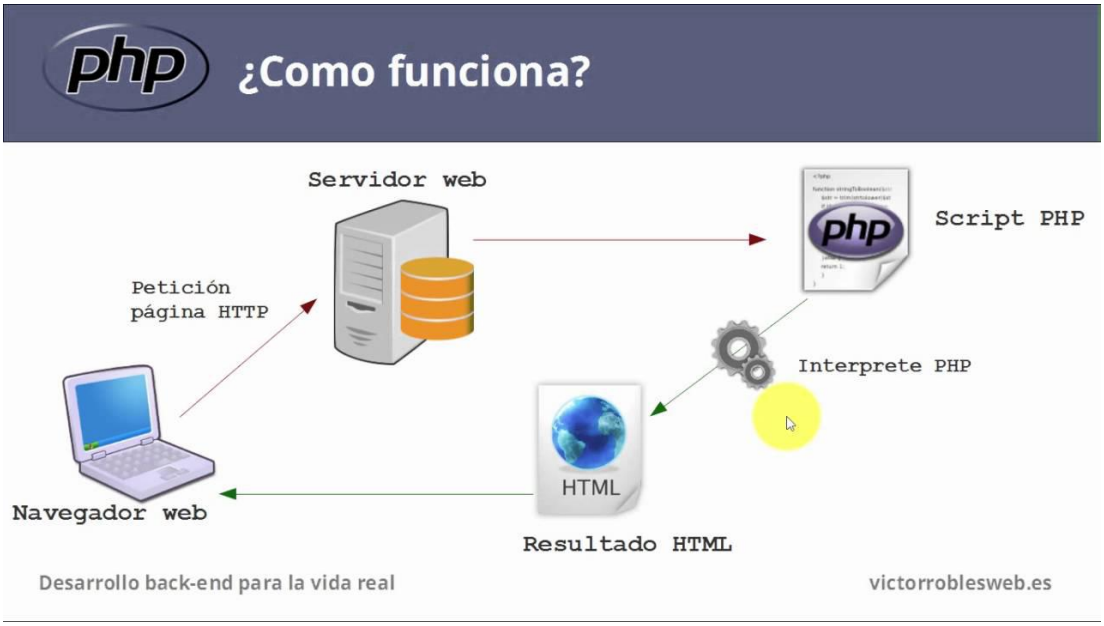
¿Qué es PHP?

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

¿Cómo funciona PHP?

- El usuario ingresa la URL en su navegador y presiona ENTER
- El navegador realiza un pedido HTTP GET al servidor
- El servidor recibe el pedido e identifica que la URL llama a un archivo .php
- El servidor invoca al intérprete PHP con el contenido del archivo
- El interprete PHP procesa todo el código que pertenece a PHP
- El interprete PHP retorna el nuevo código generado (HTML) al servidor y esté al navegador del usuario
- El navegador del usuario realiza el render del contenido HTML



Algunos sitios de ejemplo que utilizan PHP

- Facebook
- Wikipedia
- Flickr
- Yahoo
- iStock
- PhotoTumblr
- WordPress.com
- MailChimp
- PhpMyAdmin

Ventajas de utilizar PHP

- Es rápido
- Se puede correr en diferentes plataformas (Linux, Windows, OSX)
- Simple
- Es gratuito
- Muy buena documentación

Sintaxis básica de PHP

- Todo el código PHP debe ir entre **<?php** y **?>**

```
BASICA
<html>
    <body>
        <h1>Titulo</h1>
        <p>más código html</p>
        <?php
            // Entre <?php y ?> va todo el código PHP
        ?>
    </body>
</html>
```

- Lo usamos en documentos HTML con extensión **.php**
- Cada instrucción termina con **;**

```
echo "El valor de la segunda variable es:";
echo $dia2;
```

• Uso de variables y constantes en PHP

Variables

- Una variable es un espacio que podemos almacenar información, la cual podemos acceder por medio de un nombre simbólico
- En PHP las variables comienzan con el símbolo **$**
- Es sensible a mayúsculas y minúsculas **($nombre** no es lo mismo que **$Nombre**)
- Si almacenamos texto, utilizamos comillas simples o dobles para asignar la información
- El nombre debe comenzar con una letra o underscore **(_)**, luego puede números, letras, o underscores
- Para asignar un valor usamos el signo **=** (poniendo la variable a la izquierda del **=,** y el dato que queremos asignar a la derecha)

```
1 USO DE VARIABLES
2 <?php
3     $nombre = "Juan";
4     $apellido = "Gomez";
5     echo "$nombre, $apellido";
6     echo "<br>";
7     echo '$nombre, $apellido'; // Cuidado con las comillas simples
8 ?>
```

Imprimir información por pantalla en PHP

Para imprimir contenido (variables, texto, HTML) por pantalla podemos utilizarlo mediante cualquiera de las siguientes funciones:

- **Echo**
  - puede imprimir más de una separadas por coma
- **Print**
  - imprime una cadena

```
print 'Hola';
echo 'Hola', 'Hola de nuevo';
```

- **Var_dump y Print_r**
  - imprimen los detalles de una variable, incluyendo su valor, en un formato legible por el humano. Si es un array o un objeto también imprimen los detalles de cada elemento. Se utilizan frecuentemente durante la depuración de código, situación en la que var_dump suele ser más útil por la mayor información que proporciona.
  - **var_dump** proporciona información sobre el tamaño y tipo de datos de la variable y, en el caso de arrays y objetos, de los elementos que la componen. **print_r** no da información sobre el tamaño de la variable ni sobre el tipo de datos.

```php
$foo = array( 5, 0.0, "Hola", false, '' );

var_dump( $foo );
//Imprime
array(5) {
    [0]=> int(5)
    [1]=> float(0)
    [2]=> string(4) "Hola"
    [3]=> bool(false)
    [4]=> string(0) ""
}

print_r( $foo );
//Imprime
Array (
    [0] => 5
    [1] => 0
    [2] => Hola
    [3] =>
    [4] =>
)
```

Ejercicio propuesto

Crear un script php denominación **imprimir.php** en el mismo crear la variable **$mi_nombre**, asignar un string con tu nombre e imprimir el contenido de dicha variable por pantalla con echo, print, var_dump y print_r.

Constantes en PHP

- Es un espacio que podemos almacenar información, la cual podemos acceder por medio de un nombre simbólico
- Su valor no puede ser modificado (esta es la diferencia principal con las variables)
- No llevan el signo $ en su nombre, como las variables.Para su definición se utiliza.

```
<?php
define("<nombre de la constante>", "<valor de la constante>");
?>
```

- Es sensible a mayúsculas y minúsculas (nombre, no es lo mismo que NOMBRE)
- Si almacenamos texto, utilizamos comillas simples o dobles para asignar la información
- El nombre debe comenzar con una letra o underscore (_), luego puede números, letras, o underscores

Ejercicio propuesto

Dentro del script PHP utilizado en el ejercicio anterior, crear una constante _CURSO. Definir esa constante con el valor **2017** hacer un echo de dicha constante.

Intentar modificar una línea debajo el valor de la constante, ¿que nos arroja en pantalla?

Operadores aritméticos

Operador	*Descripción*	*Ejemplo*
+	Suma	5 + 3 //8
-	Resta	5 - 3 //2
*	Multiplicación	5 * 3 //15
/	División	10 / 5 //2
%	Módulo de la división	5 % 3 // 2
**	Exponenciación	5 ** 2 //25
.	Concatena cadenas de caracteres	$nombre = "Juan".", "."Pérez";

Operadores de asignación

Operador	*Descripción*	*Ejemplo*
=	El operador de la izquierda, toma el valor de la expresión de la derecha	$nombre = "Juan";
+=	Al operador de la izquierda, se le suma el valor de la expresión de la derecha (numérico)	$numero += 3
-=	Al operador de la izquierda, se le resta el valor de la expresión de la derecha (numérico)	$numero -= 3
++	Permite incrementar en 1 el operador de la izquierda (numérico)	$numero++
--	Permite decremento en 1 el operador de la izquierda (numérico)	$numero--
.=	Al operador de la izquierda, se le concatena la cadena de la derecha (cadenas)	$nombre .= ", Pérez";

Operadores de comparación

Operador	*Descripción*	*Ejemplo*
==	Si el valor de la derecha es igual al de la izquierda retorna TRUE (verdadero), sino retorna FALSE (Falso)	$edad == 33
!=	Si el valor de la derecha NO es igual al de la izquierda retorna TRUE, sino retorna FALSE	$edad != 33
<	Si el valor de la izquierda es menor al de la derecha retorna TRUE, sino retorna FALSE	$edad < 33
<=	Si el valor de la izquierda es menor o igual al de la derecha retorna TRUE, sino retorna FALSE	$edad <= 33
>	Si el valor de la izquierda es mayor al de la derecha retorna TRUE, sino retorna FALSE	$edad > 33
>=	Si el valor de la izquierda es mayor o igual al de la derecha retorna TRUE, sino retorna FALSE	$edad >= 33

Operadores lógicos

Operador	*Descripción*	*Ejemplo*
&&	Si se cumple la expresión de la izquierda y la de la derecha retorna TRUE, sino retorna FALSE	$edad == 33 && $nombre == "Juan"
||	Si se cumple la expresión de la izquierda O la de la derecha retorna TRUE, sino retorna FALSE	$edad < 33 || $edad > 40
!	Niega el resultado de la expresión (invierte su valor, si es TRUE, retorna FALSE)	! ($edad < 33)

Funciones

¿Qué son las funciones?

- Las funciones son módulos auto contenidos de código para cumplir una determinada operación
- Las funciones "encapsulan" una tarea
- Son utilizadas para tareas que utilizamos varias veces
- Las funciones reciben parámetros (variables) y retornan un resultado (que puede ser nulo/vacio)

```
FUNCIONES
<?
function <nombre de la función>(<listado de parámetros que recibe>) {
    // Código a ejecutar con los parámetros
    return <valor que retorna>;
}
?>
```

Ejemplo de función que realiza una suma

```php
<?php
/**
 * Suma dos números
 * Parametros númericos a, b
 * Retorna la suma de los números
 */
function suma($a, $b) {
    $total = $a + $b;
    return $total;
}
$total = suma(5,4);
echo "La suma de 5 y 4 es ".$total; // 9
?>
```

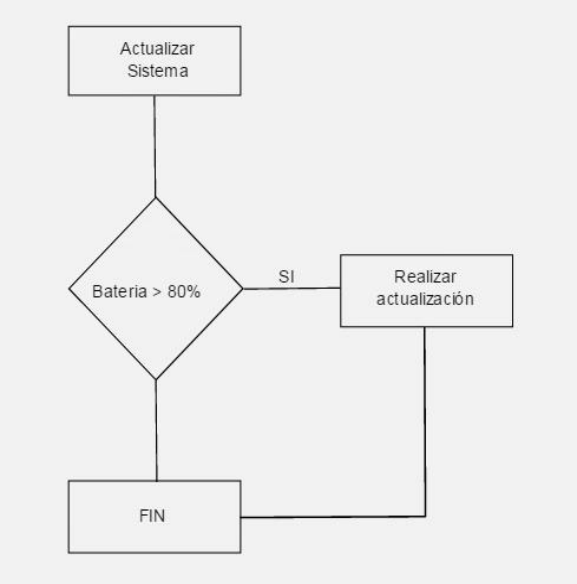
Ejercicio propuesto

Crear una función que reciba por parámetro 2 strings, los concatene y luego los imprima por pantalla.

Desarrollar la función y el llamado a la misma.

• Condicionales

Utilizamos condicionales para tomar alguna decisión en el código.



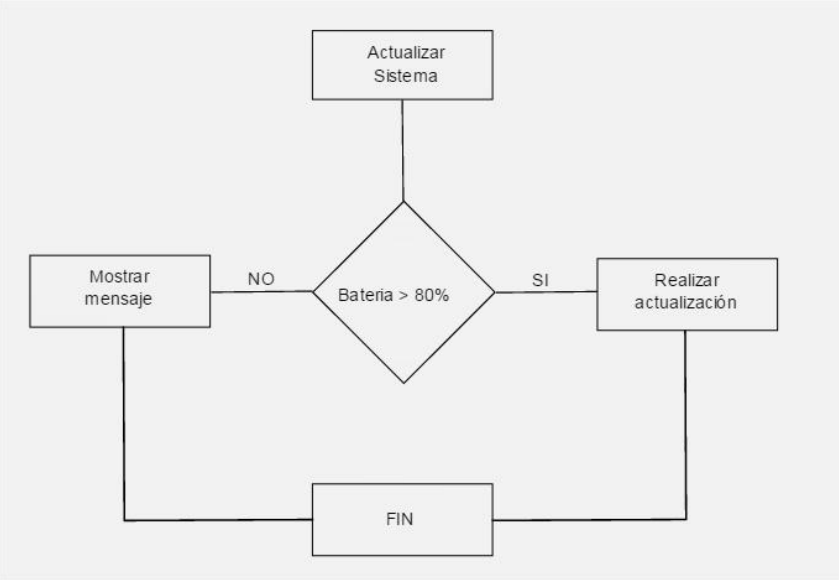
En el ejemplo anterior vemos que si la batería es mayor a un 80% entonces realizamos la actualización del sistema, este es un ejemplo de la utilización de condicionales.

Sintaxis

```
<?
if (<condiciones>) {
    // Código cuando se cumple la condición
} else {
    // Código cuando NO se cumple la condición
}
?>
```

```
1 <?
2 EJEMPLO
3 if ($edad>=18) {
4     echo "Es mayor";
5 } else {
6     echo "Es menor";
7 }
8 ?>
```

IF / ELSE

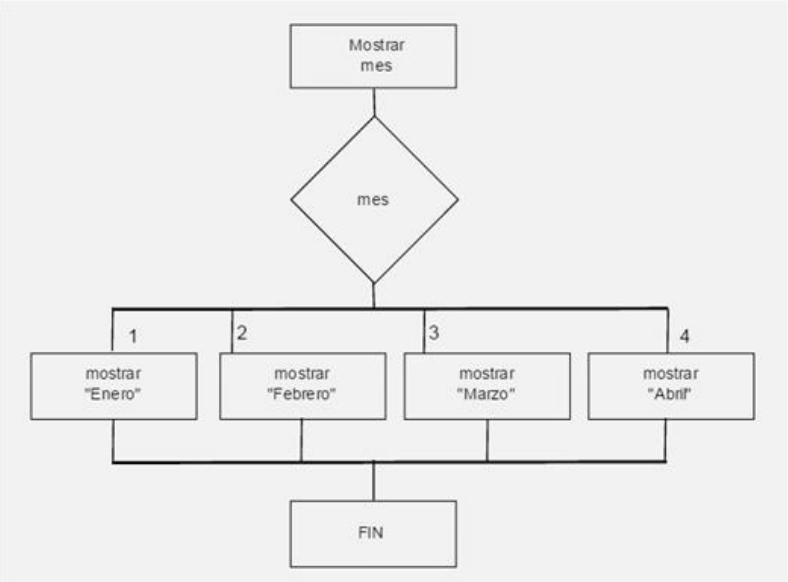


En este caso no solo tomamos una decisión (acción) si la condición se cumple, sino que también lo hacemos en caso de que no se cumpla.

Ejemplo

```php
<?php
function actualizar($bateria) {
    if ($bateria>80) {
        echo "Realizando actualización";
    } else {
        echo "Debe cargar el dispositivo";
    }
}
?>
```

SWITCH



El switch lo utilizamos cuando puede haber varias acciones de acuerdo al valor que tome nuestro ítem a comprar, en el ejemplo vemos que si el mes es 1 debemos mostrar “Enero”, 2 “Febrero”, etc.

```php
<?php
switch (<variable a analizar>) {
    case <valor a comparar>:
        // codigo
        break;
    case <otro valor a comparar>:
        // codigo
        break;
    ...
    // Opcional: si deseamos realizar una operación
    //           cuando no concuerda con las comparaciones anteriores
    default:
        // codigo
    // /Opcional
}
?>
```

```php
<?php
function mostrarMes($mes) {
    switch($mes) {
        case 1:
            echo "Enero";
            break;
        case 2:
            echo "Febrero";
            break;
        case 3:
            echo "Marzo";
            break;
        case 4:
            echo "Abril";
            break;
        default:
            echo "Mes inválido";
    }
}
?>
```

- **Ciclos**

Un ciclo es una iteración que vamos a realizar sobre un array o vector, es decir vamos a "recorrer" cada elemento de un array utilizando ciclos.

WHILE

```
WHILE
<?
while () {
      // Código que se ejecuta por cada iteración
}
?>
```

El ciclo while ejecutara el código que tiene en su cuerpo mientras la condición sea **true**.

```
1 WHILE EJEMPLO
2 <?
3 $i=1;
4 while($i<10) {
5     echo $i;
6     $i++;
7 }
8 ?>
```

En este ejemplo vemos que se ejecutara **echo $i; $i++;** mientras **$i<10**

FOR

```
<?
for (<inicialización>; <condición de fin>; <modificador por iteración>) {
        // Código que se ejecuta por cada iteración
}
?>
```

Es el ciclo más utilizado, está compuesto por 3 elementos:

- Inicialización: Vamos a introducir las variables que queremos inicializar, por ejemplo **$i=0;**
- Condición de fin: Sera la condición por la cual se cortara el ciclo, por ejemplo **$i<10**.
- Modificador por iteración: al terminar de ejecutarse la iteración se ejecutara este modificador, por ejemplo **$i++;**

El ciclo entonces, iterara mientas su condición de fin sea **true.**

```
EJEMPLO
<?
for($i=1; $i<10; $i++) {
    echo $i;
}
?>
```

En este ejemplo vemos que se inicializa la variables **$i** en 1. La condición del ciclo será **$<10** y en cada finalización de iteración se incrementara **$i** en una unidad.

El ciclo FOR es un ciclo WHILE con algunos componentes más, que nos permiten una facilidad de uso mayor.

- # Vectores

¿Qué es un vector?

Es una zona de almacenamiento contiguo que contiene una serie de elementos del mismo tipo, los elementos de la matriz.

Utilizamos un vector, cuando queremos almacenar en una variable más de un valor.

Por ejemplo queremos tener en una variable una serie de números los cuales seleccionaremos al azar y así poder jugar al loto:

```
CREAR VECTOR EJEMPLO
<?
$numeros = array(1,30,22,94);
?>
```

La variable **$numeros** tendrá todas las posibilidades a elegir.

Para acceder a una posición del vector lo haremos de la siguiente manera:

```
1 ACCEDER A DATOS DEL VECTOR
2 <?
3 $numeros[2]; // Para acceder a la posicion 3 del vector (valor 22)
4 ?>
```

LOS VECTORES COMIENZAN DESDE LA POSICION 0

Cómo está compuesto un vector

Los vectores están compuestos por 2 elementos:

- Clave o índice
  - Mediante esta identificaremos la posición del vector para poder almacenar o acceder a un valor. En el ejemplo las claves serian 0,1,2,etc
- Valor
  - Es el valor propiamente dicho, en el ejemplo 1,30,22,94

Entonces por ejemplo en el índice 0 del array **$numeros** tendremos el valor 1.

Vectores asociativos

En PHP podemos colocar como índices de un vector un string, esto es lo que se denomina un vector asociativo.

```
CREAR VECTOR ASOCIATIVO
<?
variable = array(<indice_1> => <valor_indice_1>, ..., <indice_n> => <valor_indice_n>);
?>
```

Por ejemplo:

```
1 CREAR VECTOR ASOCIATIVO EJEMPLO
2 <?
3 $datos = array("nombre" => "juan", "edad" => 33);
4 ?>
```

En este caso el array tendrá datos de una persona, cada uno de ese tipo de dato podrá estar identificado mediante su clave.

Si queremos acceder al clave nombre de la persona lo haremos de la siguiente manera:

```
ACCEDER A DATOS DE UN VECTOR ASOCIATIVO
<?
$datos["nombre"];
?>
```

Ejercicio propuesto

- Crear un script PHP con un vector vacio y completarlo con los números 1,2,3,4,5 (usando el bucle for)
- Crear un script PHP con un vector vacio y completarlo con los números 1,2,3,4,5 (usando el bucle while)

- # Formularios

¿Para que utilizamos los formularios?

Los formularios los podremos utilizar para que el usuario pueda enviar información a nuestra aplicación. Un ejemplo clásico de formularios es el formulario de registro / login o el de contacto.

Ejemplo de sintaxis HTML de un formulario

```
<form action="formulario.php" method="post">
    <input type="text" name="mensaje" />
    <input type="submit" />
</form>
```

- Action: Especifica la URL donde serán enviados los datos cuando se submite el formulario
- Method: Especifica el método HTTP utilizado al enviar los datos (GET o POST)
- name: Especifica el nombre del formulario

Método	***Disponible en variable***
GET	$_GET[<nombre del campo>]
POST	$_POST[<nombre del campo>]

Para acceder a los datos enviados desde un formulario debemos utilizar las variables $_GET y $_POST en el script PHP que recibirá los datos del mismo.

Ejercicio propuesto

Crear un formulario de registro en el archivo **formulario.html** que utilice el método POST, y definir en su action **formulario.php**.

El formulario debe contar con los siguientes campos:

- Nombre
- Apellido
- Email
- Password
- Confirmar password

Luego en el archivo **formulario.php** hacer un **var_dump($_POST)**, de acuerdo a lo observado ¿Qué es la variable $_POST?

- **Session**

Permite mantener información del usuario en el servidor, que se puede utilizar en cualquier página del sitio.

Mantienen la información de un único usuario.

Ejemplo:

- Si realizó el login
- Nombre de usuario
- Color favorito

Toda la información es almacenada en un vector asociativo especial

$_SESSION

En el cual se puede acceder, o consultar la información guardada previamente

$_SESSION["username"] = “utnalumno"; // Guardamos información ...

echo $_SESSION["username"]; // Consultamos la información guardada

Cómo iniciar sesión

Para que PHP permita operar con sesiones, se debe incluir antes de cualquier salida HTML la siguiente instrucción en PHP

session_start(); // Crea una sesión o recupera una anterior

Cuando se desea borrar la información que hay en la sesión (ejemplo: cuando el usuario sale del sistema), se utiliza:

session_unset(); // Borra todas las variables de la sesión

session_destroy(); // Borra la sesión

Ejercicio propuesto

Dado el ejercicio anterior (formulario de registro) almacenar los datos ingresados en el formulario en sesión (no olvidar hacer el **sesión_start()**) y luego hacer un **var_dump($_SESSION)** en el archivo **formulario_session.php** (tampoco olvidar hacer el **sesión_start()** aquí)

• Base de datos

Definición

Es una colección de datos organizados que permite su rápida búsqueda y recuperación.

Una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

SQL

- SQL (Foreign Key) Structured Query Language: Es un tipo de lenguaje declarativo para el acceso a bases de datos relacionales
    - Permite crear, modificar, y eliminar tablas, campos y relaciones
    - Permite guardar, recuperar, buscar, actualizar, y borrar registros
    - Permite administrar permisos de acceso a los datos

Definiciones importantes

- Tabla: Es una colección de datos relacionados que se mantienen en un formato estructurado dentro de la base de datos (Similar a una hoja en Excel)
- Campo (columna): Conjunto de valores de un mismo tipo (Similar a una columna en Excel)
- Registro (fila): Los valores que toma cada campo (Similar a una fila en Excel)
- Clave primaria: Es el campo que nos permite identificar unívocamente un registro de la misma tabla
- Clave foránea: Es el campo que nos permiten identificar unívocamente un registro de otra tabla

Aquí vemos las tablas curso y alumno.

La tabla curso tiene las columnas nombre y turno, mientras que la tabla alumno tiene las columnas nombre y edad

Un registro de ejemplo en curso seria:

- Nombre: PHP avanzando
- Turno: Noche

Es decir un registro es un ejemplo puntual formado por los tipos de datos especificados en las columnas.

- **MySql**

Definición

MySQL es un "gestor de bases de datos", es decir, una aplicación informática que se usa para almacenar, obtener y modificar datos (realmente, se les exige una serie de cosas más, como controlar la integridad de los datos o el acceso por parte de los usuarios, pero por ahora nos basta con eso).

Operaciones básicas sobre tablas en MySql

Crear tabla

Creamos la tabla en la cual almacenaremos los datos.

Debemos especificar columna y tipo de dato a crear.

```
CREATE TABLE <nombre_tabla> (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    <nombre_campo> <tipo_campo>,
    ...
    <nombre_campo> <tipo_campo>
);
```

Ejemplo

```
CREATE TABLE curso (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    turno VARCHAR(100)
);
```

Modificar tabla

Mediante esta sintaxis podremos modificar la estructura de nuestra tabla (ver phpmyadmin)

```
ALTER TABLE <nombre_tabla> DROP COLUMN <nombre_columna>;
```

Eliminar tabla

```
DROP TABLE alumno

ALTER TABLE curso DROP COLUMN codigo;
```

Eliminar todos los registros de una tabla

```
TRUNCATE alumno;
```

- **Operaciones básicas sobre registros en MySql**

Insertar datos

Sintaxis

```
INSERT INTO <nombre_tabla>
        (<columna_1>, <columna_2>, ..., <columna_n>)
VALUES  (<valor_col_1>, <valor_col_2>, ..., <valor_col_n>);
```

Ejemplo

```
INSERT INTO curso (nombre, turno) VALUES ('Webmaster 1', 'Noche');
```

Actualizar datos

Sintaxis

```
UPDATE <nombre de la tabla>
    SET <nombre de campo>=<nuevo valor>, <nombre de campo>=<nuevo valor>, ...
    WHERE <listado de condiciones
```

Ejemplo

```
UPDATE curso
    SET nombre='Webmaster introductorio'
    WHERE nombre='Webmaster 1';
```

Eliminar datos

Sintaxis

```
DELETE FROM <nombra de la tabla>
    WHERE <listado de condiciones>
```

Ejemplo

```
DELETE FROM curso
    WHERE nombre like '%introductorio%';
```

Consultar datos

Sintaxis

```
SELECT <listado de columnas separadas por coma o *>
  FROM <nombre de la tabla>
  WHERE <listado de condiciones>;
```

Ejemplo

```
SELECT nombre, id
  FROM curso
  WHERE id>3;
```

• Tipos de datos en MySql

Enteros

- **TINYINT:** Entero muy pequeño (puede tomar el valor desde -128 a 127)
- **SMALLINT:** Entero pequeño (puede tomar el valor desde -32.728 a 32.727)
- **MEDIUMINT:** Entero medio (puede tomar el valor desde -8.388.608 a 8.388.607)
- **INT:** Entero (puede tomar el valor desde -2.147.483.648 a 2.147.483.647)
- **BIGINT:** Entero grande (puede tomar el valor desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807)
- **BIT:** Booleano (puede tomar el valor 0 o 1)

Reales

- **DECIMAL:** Parte entera + decimal (Se debe especificar la cantidad total de dígitos y cuántos son decimales) Ejemplo DECIMAL(5,2) 123,45
- **FLOAT:** Para operaciones matemáticas (se debe especificar la cantidad total de dígitos y decimales).

Cadenas

- **CHAR:** Cadena con una longitud fija que se especifica (longitud máxima 255)
- **VARCHAR:** Cadena hasta la longitud que se especifica (longitud máxima 65.535)
- **TEXT:** Cadena de texto muy larga
- **BLOB:** Datos binarios

Fecha y hora

- **DATE:** Para guardar una fecha en formado AAAA-MM-DD
- **TIME:** Para guardar una hora en formato HH:MM:SS
- **DATETIME:** Para guardar una fecha y hora en formato AAAA-MM-DD HH:MM:SS

- # Operadores

Aritméticos

- **+:** Suma
- **-:** Resta
- ***:** Multiplicación
- **/:** División
- **%:** Resto de la división

Comparación

- **=:** Verifica que los valores sean iguales. 5 = 3 => False
- **!= o <>:** Verifica que los valores NO sean iguales. 5 != 3 => True
- **<:** Verifica que el valor de la izquierda sea menor al de la derecha. 5 < 3 => False
- **>:** Verifica que el valor de la izquierda sea mayor al de la derecha. 5 > 3 => True
- **<=:** Verifica que el valor de la izquierda sea menor o igual al de la derecha. 5 <= 3 => False
- **>=:** Verifica que el valor de la izquierda sea mayor o igual al de la derecha. 5 >= 3 => True

Lógicos

- **AND:** Verifica que se cumplan las dos condiciones. 2 < 3 AND 3 < 5 => True
- **BETWEEN:** Verifica que un valor esté entre otros dos. 3 BETWEEN 2 AND 5 => True
- **IN:** Verifica que un valor esté dentro de una lista de valores. 2 IN (1,2,3,4,5) => True
- **LIKE:** Verifica que un valor esté contenido en otro (string)'Jorge Luis Borges' like '%Luis%' => True
- **NOT o !:** Niega el resultado de la siguiente operación. 3 NOT BETWEEN 2 AND 5 => False

- **Funciones**
  - **MAX:** Retorna el valor máximo de los valores. MAX (20,100,180) => 180)
  - **MIN:** Retorna el valor mínimo de los valores. MIN (20,100,180) => 20
  - **SUM:** Retorna la suma de los valores. SUM (20,100,180) => 300
  - **COUNT:** Retorna la cantidad de valores. COUNT (20,100,180) => 3
  - **AVG:** Retorna el promedio entre los valores. AVG (20,100,180) => 100

- **Joins**

¿Para que utilizamos los joins?

Nos permite "unir" los resultados de dos o más tablas relacionadas.

Ejemplo: tenemos la tabla cursos y alumno, la tabla alumno tiene una relación con la tabla curso.

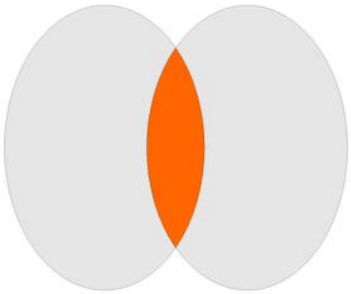
Queremos desarrollar una consulta la cual nos muestre el nombre, edad y turno al que asiste cada uno de los alumnos. Para resolver este problema debemos hacer un join entre la tabla alumno y curso y mostrar la columna turno de la tabla curso.

Tipos de joins

Tipo	Descripción
Inner Join	Los resultados están en las dos tablas relacionadas
Left Join	Los resultados están en las dos tablas relacionadas, o en la tabla izquierda
Right Join	Los resultados están en las dos tablas relacionadas, o en la tabla derecha

Inner Join

Muestra solo los resultados de ambas tablas

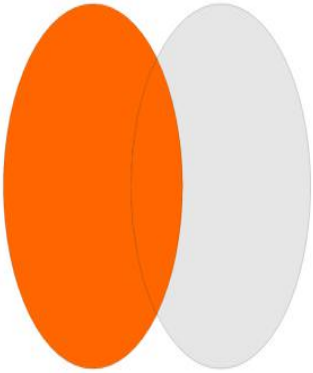


Ejemplo

```
select *
    from alumno
    INNER JOIN curso ON curso.id=alumno.curso_id
```

Left Join

Muestra solo los resultados que están en ambas tablas o en la tabla de la izquierda

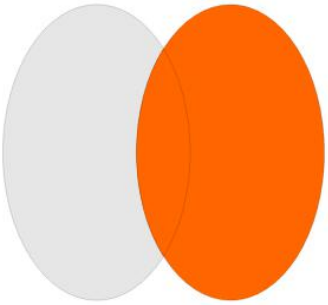


Ejemplo

```
select *
    from alumno
    LEFT JOIN curso ON curso.id=alumno.curso_id
```

Right Join

Muestra solo los resultados que están en ambas tablas o en la tabla de la derecha



Ejemplo

```
select *
    from alumno
    RIGHT JOIN curso ON curso.id=alumno.curso_id
```

- **PHPMYADMIN**

Crear base de datos

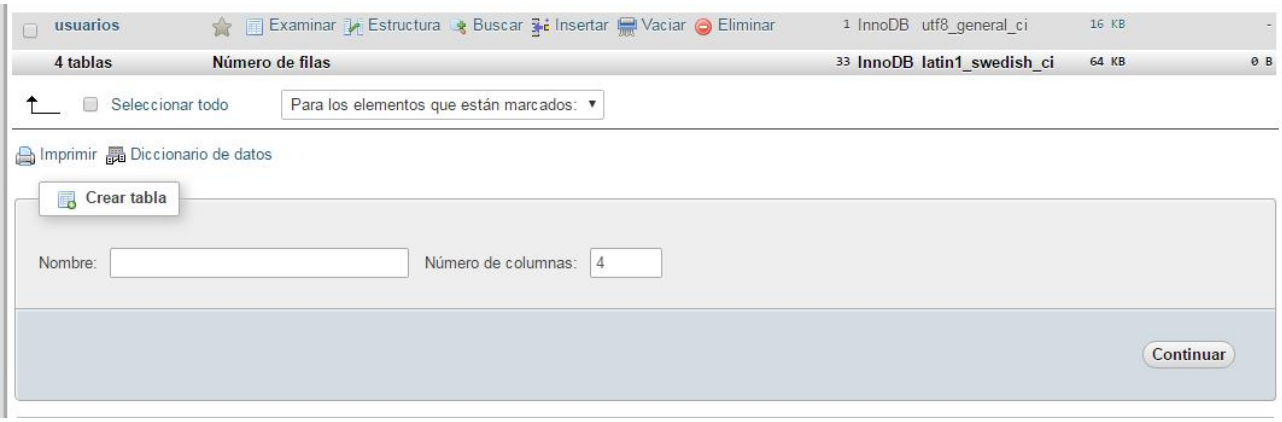
Lo primero que debemos hacer para comenzar a trabajar es crear nuestra base de datos

Debemos acceder a: http://localhost/phpmyadmin/

Colocamos el nombre de la base de datos que queremos crear y hacemos clic en “Crear”

Crear una tabla

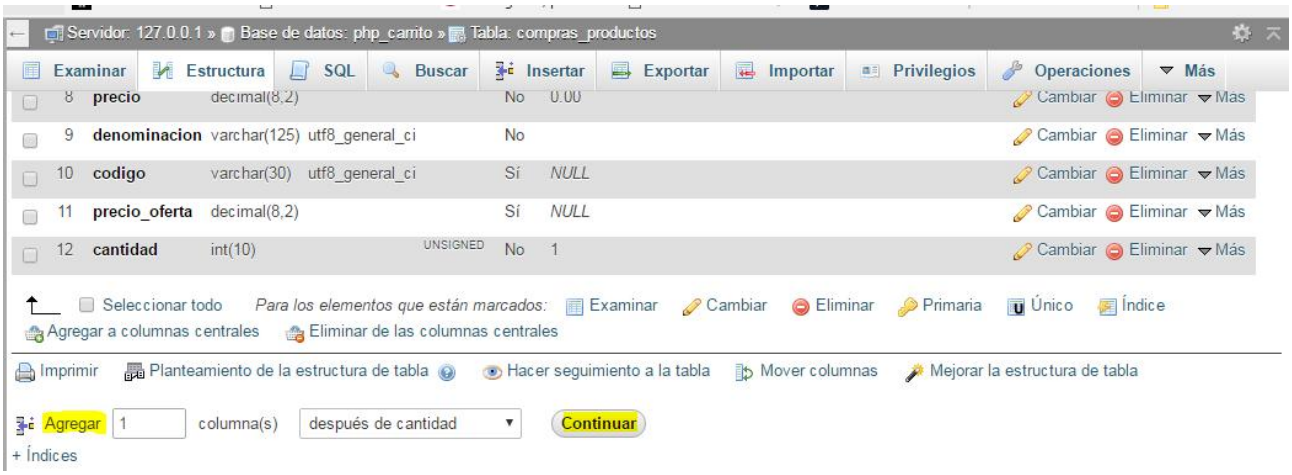
Debemos indicar nombre de la tabla y hacer clic en continuar



Luego se abrirá una pantalla para cargar 4 columnas de la misma (ver agregar columna a una tabla)

Agregar columna a una tabla

Nos ubicamos dentro de la base de datos donde queremos crear la tabla, vamos a la pestaña “Estructura”



Luego debemos definir los campos que queremos agregar:

- Nombre: Indica nombre de la columna
- Tipo: Tipo de dato (ver tipos de datos en mysql)

- Longitud: El tamaño máximo que tendrá la columna, por ejemplo en caso de ser un texto podemos indicar 50 por lo cual no se almacenaran textos más grandes.
- Predeterminado: Debemos dejar este campo como esta
- Cotejamiento: Debemos dejar este campo como esta
- Atributos:
    - Binary: Indica que solo tomara valores tipo 0 y 1
    - UNSIGNED: Indica que no se cargaran valores negativos, esto permite un mayor rango de valores positivos.
- Nulo: En caso de que el campo este chequeado, se permite almacenar esa columna con valor null.

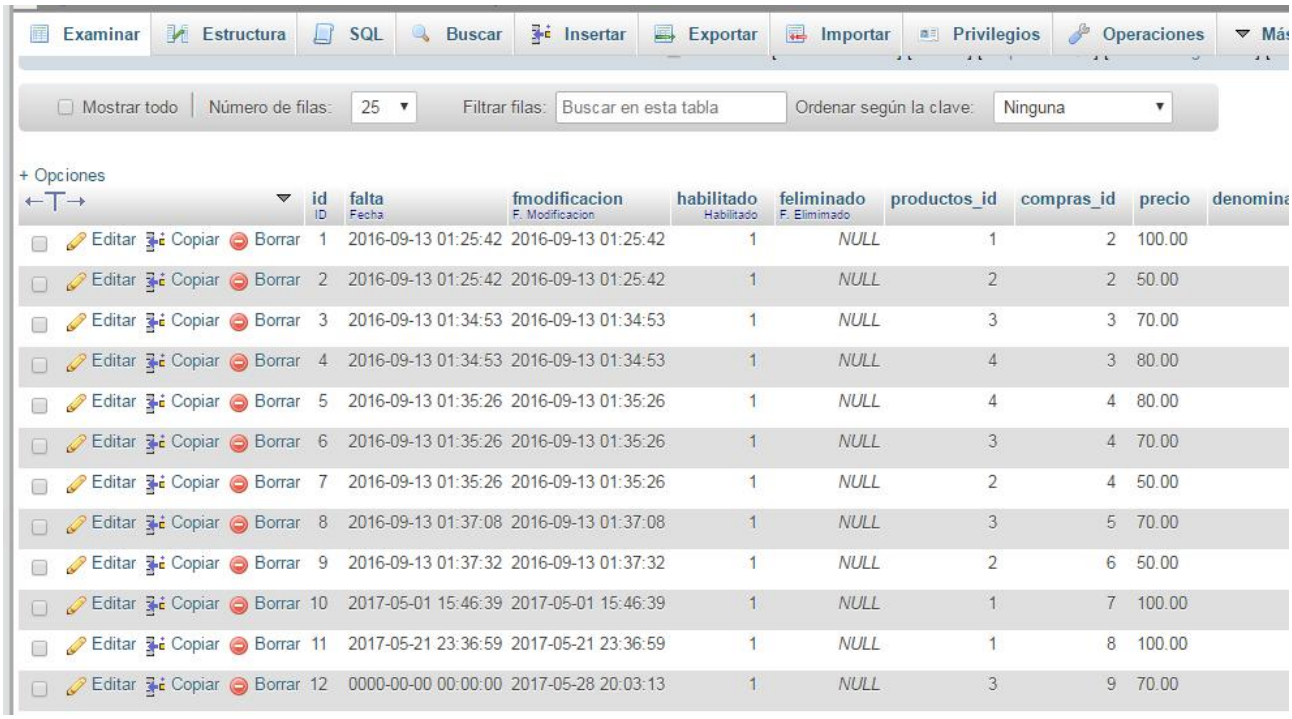
Modificar y eliminar columnas de la tabla

Para realizar modificaciones en la tabla debemos nuevamente pararnos en la pestaña "Estructura" y colocar "cambiar" para modificar la columna, o "eliminar" para eliminar la misma.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(10)		UNSIGNED	No	*Ninguna*	ID	AUTO_INCREMENT	Cambiar Eliminar Más
2	falta	datetime			No	*Ninguna*	Fecha		Cambiar Eliminar Más
3	fmodificacion	timestamp			No	CURRENT_TIMESTAMP	F. Modificacion		Cambiar Eliminar Más

Luego se nos abrirá una pantalla similar a la vista al momento de agregar una columna a la tabla

Ver los datos almacenados en una tabla

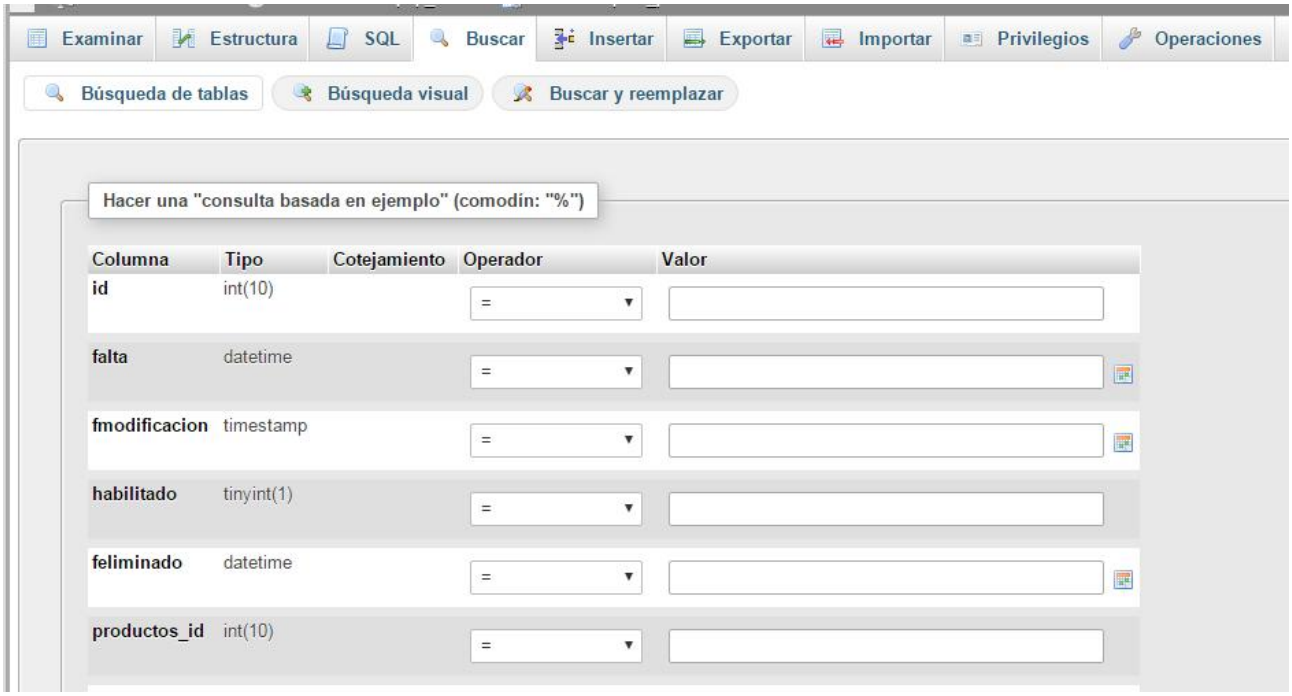


	id (ID)	falta (Fecha)	fmodificacion (F. Modificacion)	habilitado (Habilitado)	feliminado (F. Elimimado)	productos_id	compras_id	precio	denomina
Editar Copiar Borrar	1	2016-09-13 01:25:42	2016-09-13 01:25:42	1	*NULL*	1	2	100.00	
Editar Copiar Borrar	2	2016-09-13 01:25:42	2016-09-13 01:25:42	1	*NULL*	2	2	50.00	
Editar Copiar Borrar	3	2016-09-13 01:34:53	2016-09-13 01:34:53	1	*NULL*	3	3	70.00	
Editar Copiar Borrar	4	2016-09-13 01:34:53	2016-09-13 01:34:53	1	*NULL*	4	3	80.00	
Editar Copiar Borrar	5	2016-09-13 01:35:26	2016-09-13 01:35:26	1	*NULL*	4	4	80.00	
Editar Copiar Borrar	6	2016-09-13 01:35:26	2016-09-13 01:35:26	1	*NULL*	3	4	70.00	
Editar Copiar Borrar	7	2016-09-13 01:35:26	2016-09-13 01:35:26	1	*NULL*	2	4	50.00	
Editar Copiar Borrar	8	2016-09-13 01:37:08	2016-09-13 01:37:08	1	*NULL*	3	5	70.00	
Editar Copiar Borrar	9	2016-09-13 01:37:32	2016-09-13 01:37:32	1	*NULL*	2	6	50.00	
Editar Copiar Borrar	10	2017-05-01 15:46:39	2017-05-01 15:46:39	1	*NULL*	1	7	100.00	
Editar Copiar Borrar	11	2017-05-21 23:36:59	2017-05-21 23:36:59	1	*NULL*	1	8	100.00	
Editar Copiar Borrar	12	0000-00-00 00:00:00	2017-05-28 20:03:13	1	*NULL*	3	9	70.00	

En la pestaña “examinar” podemos ver todos los datos almacenados en la tabla, en cada registro podemos “editar” y “borrar” cada registro.

Buscar datos en una tabla

En la pestaña “buscar” podemos encontrar cualquier dato almacenado en la tabla



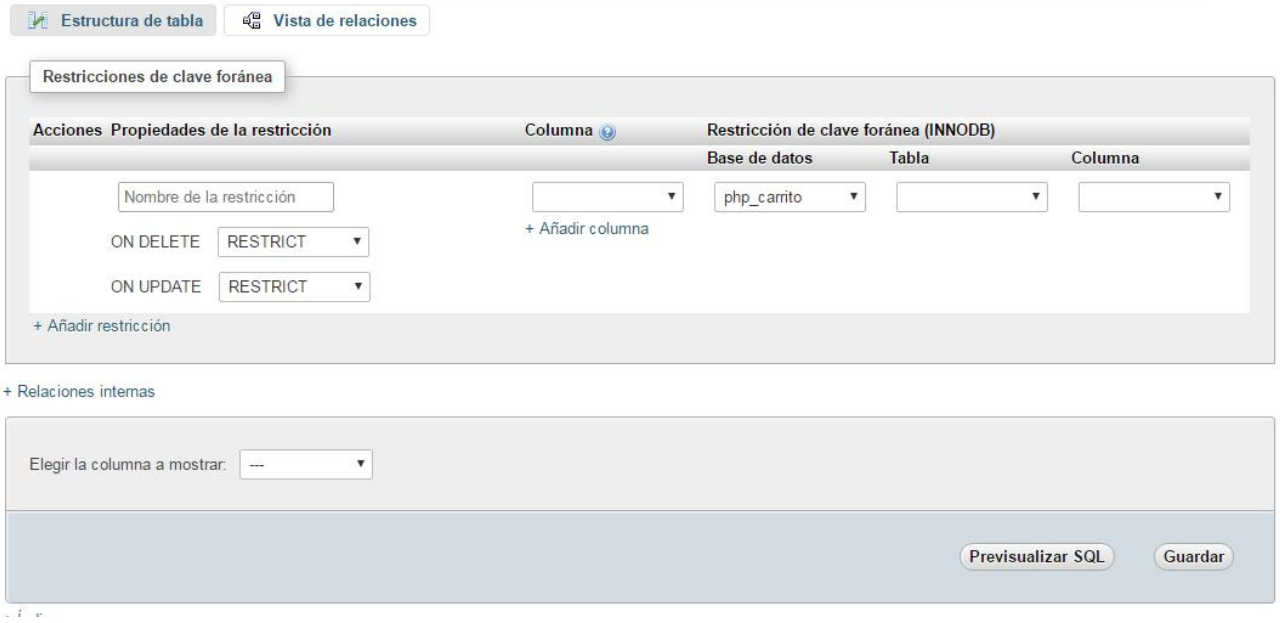
Crear una clave foránea

Para crear una tabla foránea debemos ir a “estructura” y luego a “vista de relaciones”

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(10)		UNSIGNED	No	*Ninguna*	ID	AUTO_INCREMENT	Cambiar Eliminar Más
2	falta	datetime			No	*Ninguna*	Fecha		Cambiar Eliminar Más
3	fmodificacion	timestamp			No	CURRENT_TIMESTAMP	F. Modificacion		Cambiar Eliminar Más
4	habilitado	tinyint(1)		UNSIGNED	Sí	1	Habilitado		Cambiar Eliminar Más
5	feliminado	datetime			Sí	*NULL*	F. Elimimado		Cambiar Eliminar Más
6	productos_id	int(10)		UNSIGNED	No	0			Cambiar Eliminar Más

Luego debemos indicar:

- Nombre de la restricción: Podemos colocar cualquier nombre
- Columna: Debemos seleccionar la columna de la tabla origen (tabla sobre la cual se generara la clave foránea)
- Base de datos: E elegir la base de datos destino (la cual contendrá la tabla de destino)
- Tabla: se deberá elegir la tabla de destino (es decir la tabla que tendrá la clave primaria a la cual la clave foránea hará de referencia)
- Columna: Clave primara de la tabla de destino.



Ejercicio propuesto

Crear las siguientes tablas:

- Alumnos
  - Campos
    - Id (PK)
    - Nombre
    - Apellido
    - DNI
    - Curso al que pertenece (FK a tabla cursos)
- Cursos
  - Campos
    - Id (PK)
    - Denominación
    - Turno
      - M: mañana
      - T: tarde
      - N: noche

Crear las tablas respetando el uso de Claves foráneas (FK), una vez creadas las mismas cargar con datos de pruebas sus registros, insertando, modificando y eliminando desde PHPMYADMIN.

Bibliografía utilizada y sugerida

- ¿Cómo funciona PHP?: https://www.youtube.com/watch?v=laTPz49TgZE
- https://dev.mysql.com/doc/refman/5.7/en/
- http://www.nachocabanes.com/sql/curso/sql02.php
- http://www.tutorialesprogramacionya.com/mysqlya/
- http://php.net/docs.php
- http://www.phpya.com.ar

Lo que vimos:

En esta unidad hemos visto los conceptos básicos de PHP como funciones, variables, constantes, arrays, condiciones y ciclos y MySql, los cuales nos servirán como base para los temas que veremos en la próxima unidad

Lo que viene:

En la próxima unidad veremos unos de los temas más importantes: PHP orientado a objetos.

Aprenderemos que es la Programación orientada a objetos (POO) y cómo es la sintaxis de PHP para poder programar de dicha forma.