**PASOS DE LA ARQUITECTURA**

1. Creación del MER (modelo entidad relación) que se aplicara en el proyecto



1. Creación de procedimientos almacenados (insertar, listar, consultar, editar, eliminar)



1. Creación de las carpetas de ubicación de los procesos principales

C:\xammp\htdocs\nombre del proyecto



1. Creación del archivo de conexión

Conexion.php

<?php
$conexion = mysqli\_connect ("localhost", "root", "n", "matricula") or die ("no se encontró el servidor y/o BD");
?>

**Ubicación: conexión**

1. Diseño de forma



Aspectos de interés:

Identificación del formulario



Identifica el proceso al que corresponde este formulario, en este caso es 1

**Ubicación del archivo: vista**

Se denomina un formulario independiente

Guardar con el nombre de agreagar\_alumno

**Procesos.php**

Archivo eje de todo el movimiento, por el pasan los distintos procesos, llamados, parámetros

<?php

 include $\_SERVER['DOCUMENT\_ROOT'].'/Matricula/control/clases.php';

 switch ($\_REQUEST["opc"])//url

 {

 case 1:

 $carnet = $\_POST ["carnet"];//caja de texto

 $nombre = $\_POST ["nombre"];

 alumno :: insertar ($carnet, $nombre);

 exit();

 break;}
?>

Se especifica el llamado de la clase y se evalúa la opción de referencia (opc), invoca el método con sus respectivos parámetros.

**Ubicación del archivo: control**

**Clases.php**

<?php

 class alumno

 {

 public $carnet;

 public $nombre;

 function alumno ($\_carnet, $\_nombre)

 {

 $this -> carnet = $\_carnet;

 $this -> nombre = $\_nombre;

 }

 public static function insertar ($\_carnet, $\_nombre)

 {

 include ("../conexion/conexion.php");

 $consulta = "call con\_alumno ('$\_carnet')";

 $resultado = mysqli\_query($conexion, $consulta);

 $registros = mysqli\_num\_rows($resultado);

 echo $registros;

 if ($registros == 0)

 {

 include ("../conexion/conexion.php");

 $consulta= "call ins\_alumno('$\_carnet', '$\_nombre', 'Activo')";

 $resultado = mysqli\_query ($conexion, $consulta);

 ?>

 <script language="javascript">

 alert ("Registro Almacenado")

 history.back()

 </script>

 <?php

 }

 else

 {

 ?>

 <script language="javascript">

 alert ("El Registro Ya Existe")

 history.back()

 </script>

 <?php

 }

 }

 }

Este archivo contiene la clase con sus parámetros, un constructor y el método de insertar, no contiene validaciones, invoca procedimientos almacenados.

**Ubicación: control**

**Listado de información**

Es un proceso indirecto, en el que se debe crear primero el aspecto lógico y luego la presentación, diferente al caso anterior en el que mi presentación no depende de procesos previos.

Dentro del archivo de procesos.php se adiciona un nuevo caso

case 2:

$opc = $\_REQUEST[“opc”];

$campo = (empty($\_POST[“campo”])) ? 0 : $\_POST[“campo”];

$valor = (empty($\_POST[“valor”])) ? 0 : $\_POST[“valor”];

$proceso = alumno :: listar ($opc, $campo, $valor);
include ("../vista/listar\_alumno.php");
exit();
break;

**Se adiciona el método listar (en clases.php)**

public static function listar ($\_opc, $\_campo, $\_valor)
{
include ("../conexion/conexion.php");
$mat = array ();
if ($\_opc == 2)
{
$consulta = "call lis\_alumno ()";
}
$resultado = mysqli\_query($conexion, $consulta) or die ("Consulta de Alumno es Incorrecta $consulta");
$registros = mysqli\_num\_rows ($resultado);
if ($registros == 0)
{
?>
<script language="javascript">
alert ("No Existen Registro para Mostrar")
history.back()
</script>
<?
}
else
{
for ($i = 0; $i < $registros; $i++)
{
$datos = mysqli\_fetch\_array ($resultado);
array\_push ($mat, alumno :: mostrar ($datos));
}
return $mat;
}
}

Para este método se declara un arreglo que contendrá los datos que me van a mostrar y cumplan la condición, se invoca un procedimiento o una sentencia sql que realice esta tarea, mediante un ciclo se llenan los datos del registro en el arreglo previamente creado mediante la función array\_push y se invoca el método **mostrar** que trae los datos de la tabla.

Método Mostrar.

Public static function mostrar ($datos)
{
$\_carnet = $datos ["carnet"];
$\_nombre = $datos ["nombre"];
$alumno = new alumno ($\_carnet, $\_nombre);
return $alumno;
}

Este método toma los datos de la tabla, los asigna a variables y mediante un objeto asigna y retorna a la creación de arreglo antes de visualizarlos.

Nombre listar\_alumno en carpeta vista



Diseño de la tabla donde se mostrara la información.

Dentro del código de la tabla se aplica lo siguiente:

</tr>

<?php
 foreach ($proceso as $lista)
 {
 ?>
 <tr>
 <td><?php echo $lista -> carnet; ?></td>
 <td width="282" colspan="3">&nbsp;<? echo $lista -> nombre; ?></td>
 </tr>
 <?php
 }
?>

</table>

En la última fila de la tabla se agrega la instrucción foreach, función que permite el manejo de arreglos, proceso es la contiene la matriz con la información resultante y se asigna a una variable lista que permitirá visualizar registro por registro.

La instrucción <?php echo $lista -> nombre; ?>, imprime mediante la variable lista el nombre que tenemos en la tabla, así para cada campo.

Por ser un proceso indirecto, este se invoca de una forma distinta a la tradicional, sea desde un menú o una línea directa en la barra de dirección utilizamos ../control/Procesos.php?opc=2

DISEÑAR LA SIGUIENTE TABLA PARA MANEJAR EL MENU



Guardarla en vista con el nombre de menu\_alumnos.php





Abrir uno de los navegadores en la dirección digitar localhost/matricula(nombre del proyecto)



**Inicio de variables** $

**$nombre de la variable**

Ejemplo

**cálculos.php**

<?php

$valor1 = 10;

$valor2 = 40;

$total = $valor1 + $valor2;

echo "el total es " . $total;

?>

**Bloques de control**

**if** (){

}

else {

}

Ejemplo

**Preguntas.php**

<? php

$a = 10;

$b = 40;

$total = $a + $b;

echo "el total es " . $total;

if ($a > $b) {

 echo "a es mayor que b" . "<br />\n" ;

 //. "<br />\n"

 // $b = $a;

}

else {

echo "b es mayor que a" . "<br />\n";

}

?>

**for**(){

}

Ejemplo

**Para.php**

<?php

for ( $i = 1 ; $i <= 10 ; $i ++) {

print $i."<br/>";

}

?>

**while** (condicion){

}

Ejemplo

**Mientras.php**

<? php

#EjemploDe.COM

#Ejemplo de Bucle While en PHP

$i = 0; //Inializamos la variable en 0

while($i==10) {

$i++; //Aumentamos $i en uno

echo "i ahora equivale a ".$i."</br> "; //Mostramos texto

}

echo "Fin del bucle while. El bucle while terminó en ".$i."";

?>

**switch** (){

case 1:

break;

case 2:

break;

}

Ejemplo

**Casos.php**

<?php

$i=2;

if ($i == 0) {

 echo "i es igual a 0";

} elseif ($i == 1) {

 echo "i es igual a 1";

} elseif ($i == 2) {

 echo "i es igual a 2"."<br/>";

}

switch ($i) {

 case 0:

 echo "i es igual a 0";

 break;

 case 1:

 echo "i es igual a 1";

 break;

 case 2:

 echo "i es igual a 2";

 break;

}

?>

**Investigar para la próxima clase….**

**mysqli\_connect**

Con este comando se abre una nueva conexión con Msql [1].

**mysqli\_query**

Permite realizar consultas a la base de datos [1].

**mysqli\_num\_rows**

Obtiene el número de filas de un conjunto de resultados resultado [1]

**mysqli\_affected\_rows**

Obtiene el número de filas afectadas en la última operación Mysql.

**mysqli\_fetch\_array**

Recupera una fila de resultados como un array asociativo, un array numérico o como ambos.

**include**

La sentencia **include** incluye y evalúa el archivo especificado**.**

**history.back**

Con este método le ordenas al navegador que vaya a la página que visitaste antes de llegar a esta [2].

**history.go**

Con este método le indicas al navegador que retroceda o avance tantas páginas como le indiques. SI quieres que vaya páginas hacia atrás pon valores negativos entre los paréntesis, los valores positivos te envían hacia adelante. Si pones 0 (cero) el navegador actualiza la página.

Tres páginas hacia atrás history.go(-3)

Dos páginas hacia adelante history.go(2)

Actualizar la página history.go(0)

[2]

**array\_push**

Permite Insertar uno o más elementos al final de un array o vector [1].

**[1]** [**http://www.php.net/manual/es/mysqli.query.php**](http://www.php.net/manual/es/mysqli.query.php)

**[2] http://www.paratodos.8m.net/history.htm**

**Modificación de información**

Aprovechando los recursos del sistema, tomamos el listar como base de trabajo y agregamos un link o un gráfico que nos permita seleccionar el registro especifico.

Para esto modificamos el archivo de listar\_Alumno, dándole la siguiente apariencia…



En este caso se agregó una M para representar la Modificación y una E para representar la Eliminación

Seleccionando la M se agrega el siguiente vínculo

../control/Procesos.php?opc=4&valor=<? echo $lista -> carnet; ?>

En el representamos la opción 4 para buscar el registro con el valor = carnet, y mostrarlo en el formulario respectivo antes de editarlo.

En el archivo procesos agregamos la opción correspondiente.

 case 4:

 $opc = $\_REQUEST["opc"];

 $campo = (empty($\_POST["campo"])) ? 0 : $\_POST["campo"];

 $valor = $\_REQUEST["valor"];

 $proceso = alumno :: listar ($opc, $campo, $valor);

 include ("../vista/modificar\_alumno.php");

 exit();

 break;

En el archivo clases.php en la función listar agregar la pregunta de la opción 4

elseif ($\_opc == 4)

{

 $consulta = "select \* from alumno where carnet = '$\_valor' and estado = 'activo'";

}

Quedando de la siguiente manera

if ($\_opc == 2)

 {

 $consulta = "call lis\_alumno ()";

 }

 elseif ($\_opc == 3)

 {

 if ($\_campo == 'carnet')

 {

 $consulta = "select \* from alumno where $\_campo = '$\_valor' and estado = 'activo'";

 }

 else

 {

 $consulta = "select \* from alumno where $\_campo like '%$\_valor%' and estado = 'activo'";

 }

 }

 elseif ($\_opc == 4){

 $consulta = "select \* from alumno where carnet = '$\_valor' and estado = 'activo'";

 }

Se crea al formulario de modificación que se almacena en la carpeta vista



<?php echo $listar -> carnet;?>

Para este formulario se realizan algunas modificaciones, se crea un foreach que trae la información y se muestran los datos en cajas de texto, después del body y antes del formulario

<?php
foreach ($proceso as $listar)
{
?>

Y despues de cerrar el formulario

<?php

}

?>

<form id="form1" name="form1" method="post" action="../control/procesos.php?opc=5">

Se invoca la opción 5 que es la que almacena los datos en la tabla mediante un procedimiento almacenado

Y se agrega un value en cada caja de texto que se desea modificar

<input name="carnet" type="text" id="carnet" **value="<?php echo $listar -> carnet;?>"** size="20" maxlength="10" />

Con este proceso se consulta sobre el formulario, falta el proceso 5 que haga la actualización en la tabla.

Archivo procesos.

 case 5:

 $carnet = $\_POST["carnet"];

 $nombre = $\_POST["nombre"];

 $Proceso = alumno :: editar ($carnet, $nombre);

 exit();

 break;

Archivo de clases y su método editar

Public static function editar ($\_carnet, $\_nombre)

 {

 include ("../conexion/conexion.php");

 $consulta = "call edi\_alumno ('$\_carnet', '$\_nombre')";

 $resultado = mysqli\_query($conexion, $consulta);

 $registros = mysqli\_affected\_rows($conexion);

 if ($registros == 0)

 {

 ?>

 <script language="javascript">

 alert ("Registro no se Actualizo")

 history.back()

 </script>

 <?php

 }

 else

 {

 ?>

 <script language="javascript">

 alert ("El Registro Actualizado")

 history.back()

 </script>

 <?php

 }

 }

Acá encontramos los parámetros que pasamos para afectar la modificación, igualmente los utilizamos en el procedimiento almacenado y se usa la función mysqli\_affected\_rows con el fin de conocer los registros que se modificaron.

**Eliminar información**

Se realiza un vínculo a la E del listar así

../control/Procesos.php?opc=6&valor=<?php echo $lista -> carnet; ?>

Se hace referencia a la opción 6 que será el proceso que elimine el registro (lógicamente) y se realiza el método de eliminar.

Public static function eliminar ($\_valor)

 {

 include ("../conexion/conexion.php");

 $consulta = "call eli\_alumno ('$\_valor')";

 $resultado = mysqli\_query($conexion, $consulta);

 $registros = mysqli\_affected\_rows($conexion);

 if ($registros == 0)

 {

 ?>

 <script language="javascript">

 alert ("Registro no se Elimino")

 history.go(-2)

 </script>

 <?php

 }

 else

 {

 ?>

 <script language="javascript">

 alert ("El Registro se Elimino")

 history.go(-2)

 </script>

 <?php

 }

 }

Agregar en procesos

case 6:

 $valor = $\_REQUEST["valor"];

 $Proceso = alumno :: eliminar ($valor);

 exit();

 break;