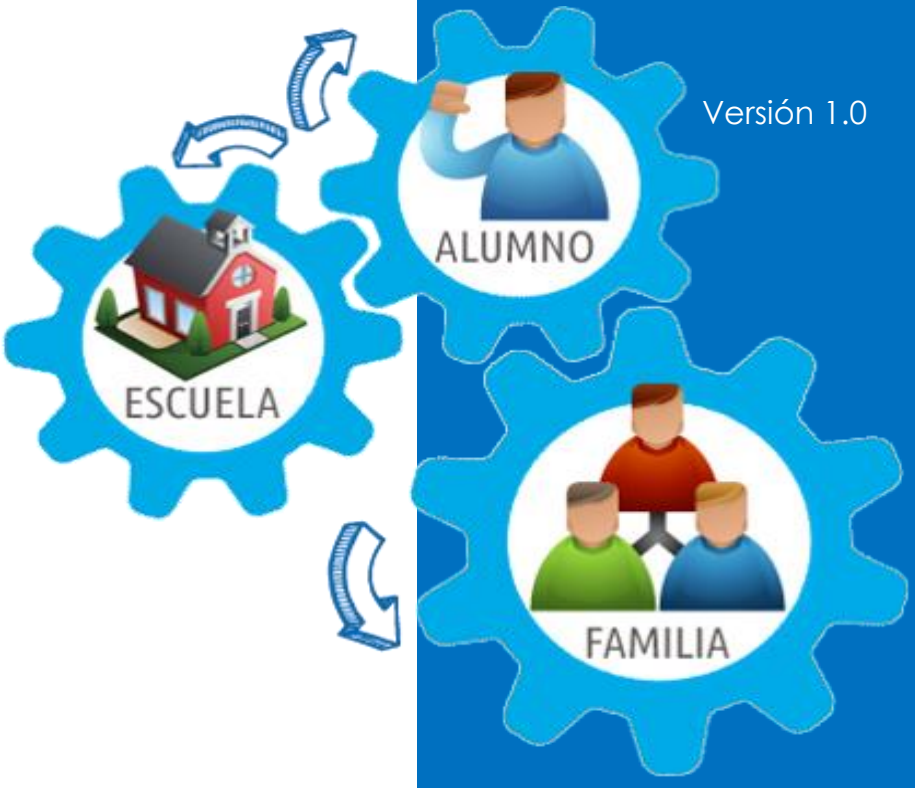
MANUAL TÉCNICO

Universidad Politécnica de Victoria

Fecha de elaboración:
23-Junio-2018

DANZLIFE - SISTEMA DE ENVÍO DE PAGOS



Versión 1.0

TECNOLOGÍA Y APLICACIONES WEB

DESARROLLADOR:
BRIAN ELÍ BECERRA HERNÁNDEZ

Especificaciones Técnicas

Sistema Operativo: Cualquiera. Acceso mediante navegador web.

Manejador de Base de Datos: MySQL.

Lenguajes: PHP, HTML, CSS, JavaScript.

Servidor de Aplicaciones: Apache.

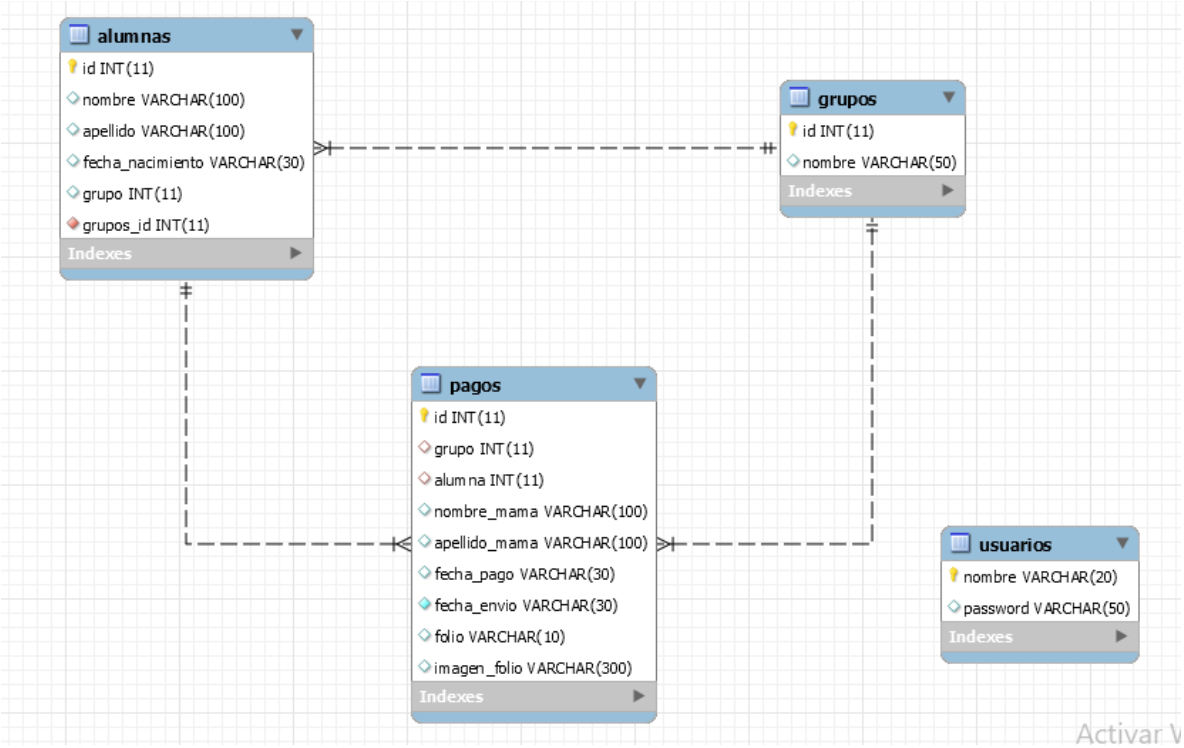
Base de datos

La información del sitio se guardará en la siguiente base de datos que contiene la siguiente estructura.

Descripción de las tablas de la base de datos:

- **alumnas:** Para guardar la información de las alumnas inscritas.
- **grupos:** Para gurdar la información de los grupos.
- **pagos:** Para guardar la información de los pagos realizados por las madres de las alumnas.
- **usuarios:** Su única función es guardar los datos de ingreso del administrador del sistema.

Diagrama de la base de datos:



Tablas de la base de datos:

Alumnas	
Columna	**Tipo**
id	int
nombre	varchar(100)
apellido	varchar(
fecha_nacimiento	varchar(30)
grupo	int

Grupos	
Columna	**Tipo**
id	int
nombre	varchar(50)

Pagos	
Columna	**Tipo**
id	int
grupo	int
alumna	int
nombre_mama	varchar(100)
apellido_mama	varchar(100)
fecha_pago	varchar(30)
fecha_envio	varchar(30)
folio	varchar(10)
imagen_folio	varchar(300)

Usuarios	
Columna	**Tipo**
nombre	varchar(20)
password	50

Descripción de la aplicación

Nombre	Ubicación	Descripción
index.php	/danzlife/public/	Este archivo sirve como plantilla para mostrar todo el contenido del sistema.
.htaccess	/danzlife/public/	Sirve para controlar el acceso al directorio *public*
config.php	/danzlife/app/config/	Contiene la información principal de la aplicación, como la ruta de la aplicación, nombre del sitio, acceso a la base de datos, etc.
alumnas.php	/danzlife/app/controllers/	Controlador para el módulo de alumnas.
grupos.php	/danzlife/app/controllers/	Controlador para el módulo de grupos.
home.php	/danzlife/app/controllers/	Controlador para el módulo principal.
App.php	/danzlife/app/core/	Contiene las funciones para transformar las url's.
Controller.php	/danzlife/app/core/	Controlador principal que comunica con las vistas.
AlumnasModel.php	/danzlife/app/models/	Modelo para el módulo de alumnas.
GruposModel.php	/danzlife/app/models/	Modelo para el módulo de grupos.
HomeModel.php	/danzlife/app/models/	Modelo para el módulo principal.
connection.php	/danzlife/app/models/	Realiza la conexión con la base de datos.
agregar.php	/danzlife/app/views/alumnas/	Vista que contiene el formulario de registro de alumnas.
editar.php	/danzlife/app/views/alumnas/	Vista que contiene el formulario para editar una alumna.
index.php	/danzlife/app/views/alumnas/	Vista que contiene la tabla de alumnas.
agregar.php	/danzlife/app/views/grupos/	Vista que contiene el formulario de registro de grupos.
editar.php	/danzlife/app/views/grupos/	Vista que contiene el formulario para editar un grupo.
index.php	/danzlife/app/views/grupos/	Vista que contiene la tabla de grupos.
admin.php	/danzlife/app/views/home/	Vista que contiene el formulario de inicio de sesión.
index.php	/danzlife/app/views/home/	Vista que el formulario de registro de pago.
lugares.php	/danzlife/app/views/home/	Vista que contiene la visualización pública de la tabla de pagos.

pagos.php	/danzlife/app/views/home/	Vista que contiene la visualización del administrador de la tabla de pagos.
footer.php	/danzlife/app/views/	Contiene el diseño del footer del sitio.
header.php	/danzlife/app/views/	Contiene el diseño del header del sitio.
.htaccess	/danzlife/app/	Sirve para controlar el acceso al directorio *app*
init.php	/danzlife/app/	Hace la importación de los archivos principales del sitio.

Funcionamiento de la aplicación

Registro de pago

Una vez que el formulario se llena, éste es enviado a una función llamada *agregarPago()* ubicada en el controlador home.

```
  //Página agregar un pago
  public function agregarPago(){
    //Es true cuando un post se envia, en este caso cuando se envía el formulario de registro
    if($_SERVER["REQUEST_METHOD"] == "POST"){
      $datos = [
        "grupo" => trim($_POST["grupo"]),
        "alumna" => trim($_POST["alumna"]),
        "nombre_mama" => trim($_POST["nombre_mama"]),
        "apellido_mama" => trim($_POST["apellido_mama"]),
        "fecha_pago" => trim($_POST["fecha_pago"]),
        "folio" => trim($_POST["folio"]),
        "nombre_img" => $_FILES["folio_img"]["name"],
        "ruta_img" => $_FILES["folio_img"]["tmp_name"]
      ];

      $this->modelo->agregarPago($datos); //Se agrega el pago

      $this->lugares(); //Se regresa al index
    }
}
```

La función guarda los datos en un arreglo y lo envía a la función *agregarPago($datos)* ubicada en el modelo home.

```
//Método para agregar un pago
public function agregarPago($data){
  $this->trans->query("INSERT INTO pagos VALUES(null, :grupo, :alumna, :nombre_mama, :apellido_mama, :fecha_pago, :fecha_envio, :folio,

  $fecha_envio = date("d-m-Y H:i:s"); //Se obtiene la fecha actual
  $destino_img = substr(RUTA_APP, 0, -4) . "/public/img/" . $data["nombre_img"]; //Se crea el destino de la imagen
  copy($data["ruta_img"], $destino_img); //Se copia la imagen al servidor

  $this->trans->bind(":grupo", $data["grupo"]);
  $this->trans->bind(":alumna", $data["alumna"]);
  $this->trans->bind(":nombre_mama", $data["nombre_mama"]);
  $this->trans->bind(":apellido_mama", $data["apellido_mama"]);
  $this->trans->bind(":fecha_pago", $data["fecha_pago"]);
  $this->trans->bind(":fecha_envio", $fecha_envio);
  $this->trans->bind(":folio", $data["folio"]);
  $this->trans->bind(":folio_img", $destino_img);

  $res = $this->trans->execute();
}
```

Esta función se encarga de guardar los datos en la base de datos.

Una vez que se haya guardado el pago en la base de datos, el controlador home redireccionará hacia la página de pagos.

Visualización de lugares (tabla de lugares)

El controlador home se encarga de direccionar el sitio hacia la vista en donde se encuentra la tabla (la página lugares), y le envía como parámetros un arreglo con los datos de los pagos, los cuales se obtienen mediante otra función ubicada en el modelo home.

```php
//Método para ir a la página para visualizar los lugares de registros
public function lugares(){
  $this->view("home/lugares", $this->modelo->getPagos());
}
```

La función del modelo se encarga de hacer una consulta a la base de datos en la que se obtienen los datos de los pagos. Una vez obtenidos, retorna un arreglo con los pagos.

```php
//Método para obtener todos los pagos
public function getPagos(){
  $this->trans->query("SELECT p.id, g.nombre, CONCAT(a.nombre, ' ', a.apellido), CONCAT(p.nombre_mama, ' ', p.apellido_mama), p.fecha_p
  $res = $this->trans->execute();

  return $res->fetchAll();
}
```

Después, en la vista de lugares se obtiene la información del arreglo pasado y mediante un ciclo se muestran los datos en forma de tabla.

```php
<!-- Se verifica que el arreglo de alumnos no esté vacío -->
<?php if(!empty($data)):?>
    <table>
      <thead>
        <tr>
          <!-- Datos de los alumnos -->
          <th>Lugar</th>
          <th>Grupo</th>
          <th>Nombre alumna</th>
          <th>Nombre mama</th>
          <th>Fecha de pago</th>
          <th>Fecha de envío</th>
          <th>Folio</th>
          <th>Comprobante de folio</th>
        </tr>
      </thead>
      <tbody>
        <!-- Ciclo para imprimir todas las lugares y su respectiva información -->
        <?php foreach($data as $pago): ?>
            <tr>
              <td><?php echo $pago[0] ?></td>
              <td><?php echo $pago[1] ?></td>
              <td><?php echo $pago[2] ?></td>
              <td><?php echo $pago[3] ?></td>
              <td><?php echo $pago[4] ?></td>
              <td><?php echo $pago[5] ?></td>
              <td><?php echo $pago[6] ?></td>
              <td><a href="<?php echo RUTA_URL . substr($pago[7], 56) ?>">Ver</a> </td>
            </tr>
          <?php endforeach ?>
```

Visualización de alumnas (tabla de alumnas)

El controlador alumnas se encarga de direccionar el sitio hacia la vista en donde se encuentra la tabla (la página alumnas), y le envía como parámetros un arreglo con los datos de las alumnas, los cuales se obtienen mediante otra función ubicada en el modelo alumnas.

```
//Página index
public function index(){
  $this->view("alumnas/index", $this->modelo->getAlumnas());
}
```

La función del modelo se encarga de hacer una consulta a la base de datos en la que se obtienen los datos de las alumnas. Una vez obtenidos, retorna un arreglo con las alumnas.

```
//Método para obtener todas las alumnas
public function getAlumnas(){
  $this->trans->query("SELECT a.id, a.nombre, a.apellido, a.fecha_nacimiento, g.nombre FROM alumnas a INNER JOIN grupos g ON a.grupo=g.
  $res = $this->trans->execute();

  return $res->fetchAll();
}
```

Después, en la vista de alumnas se obtiene la información del arreglo pasado y mediante un ciclo se muestran los datos en forma de tabla, no sin antes comprobar que haya datos existentes.

```
<!-- Se verifica que el arreglo de alumnas no esté vacío -->
<?php if(!empty($data)): ?>
    <table>
      <thead>
        <tr>
          <!-- Datos de las alumnas -->
          <th>Nombre</th>
          <th>Fecha de nacimiento</th>
          <th>Grupo</th>
          <th></th>
        </tr>
      </thead>
      <tbody>
        <!-- Ciclo para imprimir todas las alumnas y su respectiva información -->
        <?php foreach($data as $alumna): ?>
            <tr>
              <td><?php echo $alumna[1] . " " . $alumna[2] ?></td>
              <td><?php echo $alumna[3] ?></td>
              <td><?php echo $alumna[4] ?></td>
              <td>
                <!-- Botones de modificar y eliminar -->
                <a href="<?php echo RUTA_URL ?>/public/alumnas/editar/<?php echo $alumna[0] ?>" class="button radius tiny seconda
                <button name="eliminar" value="<?php echo $alumna[0] ?>" class="button radius tiny secondary">Eliminar</button>
              </td>
            </tr>
          <?php endforeach ?>
      </tbody>
    </table>
  <?php else: ?>
    <p>No hay registros</p> <!-- En caso de que no haya ningún registro en la base de datos se mostrará este mensaje -->
  <?php endif ?>
```

Registro de una alumna

Una vez que el formulario se llena, éste en enviado a una función llamada *agregar()* ubicada en el controlador alumnas.

```
//Página agregar alumna
public function agregar(){
  //Es true cuando un post se envia, en este caso cuando se envía el formulario de registro
  if($_SERVER["REQUEST_METHOD"] == "POST"){
    $datos = [
      "nombre" => trim($_POST["nombre"]),
      "apellido" => trim($_POST["apellido"]),
      "fecha_nacimiento" => trim($_POST["fecha_nacimiento"]),
      "grupo" => trim($_POST["grupo"])
    ];

    $this->modelo->agregarAlumna($datos); //Se agrega la alumna

    $this->index(); //Se regresa al index
  }
  else{
    $this->view("alumnas/agregar", $this->modelo->getGrupos());
  }
}
```

La función guarda los datos en un arreglo y lo envía a la función *agregarAlumna($datos)* ubicada en el modelo alumna.

```
//Método para agregar una alumna
public function agregarAlumna($data){
  $this->trans->query("INSERT INTO alumnas VALUES(null, :nombre, :apellido, :fecha_nacimiento, :grupo)");

  $this->trans->bind(":nombre", $data["nombre"]);
  $this->trans->bind(":apellido", $data["apellido"]);
  $this->trans->bind(":fecha_nacimiento", $data["fecha_nacimiento"]);
  $this->trans->bind(":grupo", $data["grupo"]);

  $res = $this->trans->execute();
}
```

Esta función se encarga de guardar los datos en la base de datos.

Una vez que se haya guardado el grupo en la base de datos, el controlador alumna redireccionará hacia la página donde está la tabla de alumnas.

Modificación de una alumna

Una vez que el formulario se llena, éste es enviado a una función llamada edita*r()* ubicada en el controlador alumnas.

```
//Página editar una alumna
public function editar($id = ""){
  //Es true cuando un post se envia, en este caso cuando se envía el formulario
  if($_SERVER["REQUEST_METHOD"] == "POST"){
    $datos = [
      "id" => trim($_POST["id"]),
      "nombre" => trim($_POST["nombre"]),
      "apellido" => trim($_POST["apellido"]),
      "fecha_nacimiento" => trim($_POST["fecha_nacimiento"]),
      "grupo" => trim($_POST["grupo"])
    ];

    $this->modelo->editarAlumna($datos); //Se edita la alumna

    $this->index(); //Se regresa al index
  }
  else{
    $this->view("alumnas/editar", [$this->modelo->getAlumna($id), $this->modelo->getGrupos()]);
  }
}
```

La función guarda los datos en un arreglo y lo envía a la función *editarAlumna($datos)* ubicada en el modelo alumna.

```
//Método para editar una alumna
public function editarAlumna($data){
  $this->trans->query("UPDATE alumnas SET nombre=:nombre, apellido=:apellido, fecha_nacimiento=:fecha_nacimiento, grupo=:grupo WHERE id

  $this->trans->bind(":id", $data["id"]);
  $this->trans->bind(":nombre", $data["nombre"]);
  $this->trans->bind(":apellido", $data["apellido"]);
  $this->trans->bind(":fecha_nacimiento", $data["fecha_nacimiento"]);
  $this->trans->bind(":grupo", $data["grupo"]);

  $res = $this->trans->execute();
}
```

Esta función se encarga de actualizar los datos de la alumna en la base de datos.

Una vez que se haya guardado la alumna en la base de datos, el controlador alumna redireccionará hacia la página donde está la tabla de alumnas.

Eliminación de una alumna

Al hacer clic en el botón para eliminar, se hace una llamada a la función *eliminar()* ubicada en el controlador de alumnas. Ésta hace una llamada a la función *eliminarAlumna()* ubicada en el modelo de alumnas, y le envía el id de la alumna a eliminar.

```
//Método para eliminar una alumna
public function eliminar($id = ""){

  $this->modelo->eliminarAlumna($id); //Se elimina a la alumna

  $this->index(); //Se regresa al index
}
```

Dentro de la función del modelo, se hace una consulta en la base de datos, la cual elimina a la alumna que tenga el id pasado a la función. Después de eso, el controlador redireccionará a la página de inicio de alumnas.

```
//Método para eliminar una alumna
public function eliminarAlumna($id){
  $this->trans->query("DELETE FROM alumnas WHERE id=:id");

  $this->trans->bind(":id", $id);

  $res = $this->trans->execute();
}
```

Visualización de grupos (tabla de grupos)

El controlador grupos se encarga de direccionar el sitio hacia la vista en donde se encuentra la tabla (la página de grupos), y le envía como parámetros un arreglo con los datos de los grupos, los cuales se obtienen mediante otra función ubicada en el modelo grupos.

```
//Página index
public function index(){
  $this->view("grupos/index", $this->modelo->getGrupos());
}
```

La función del modelo se encarga de hacer una consulta a la base de datos en la que se obtienen los datos de los grupos. Una vez obtenidos, retorna un arreglo con los grupos.

```
//Método para obtener todas las grupos
public function getGrupos(){
  $this->trans->query("SELECT * FROM grupos");
  $res = $this->trans->execute();

  return $res->fetchAll();
}
```

Después, en la vista de grupos se obtiene la información del arreglo pasado y mediante un ciclo se muestran los datos en forma de tabla, no sin antes comprobar que haya datos existentes.

```
<!-- Se verifica que el arreglo de alumnos no esté vacío -->
<?php if(!empty($data)): ?>
    <table>
      <thead>
        <tr>
          <!-- Datos de los alumnos -->
          <th width="500">Nombre</th>
          <th></th>
        </tr>
      </thead>
      <tbody>
        <!-- Ciclo para imprimir todas las grupos y su respectiva información -->
        <?php foreach($data as $grupo): ?>
            <tr>
              <td><?php echo $grupo[1] ?></td>
              <td>
                <!-- Botones de modificar y eliminar -->
                <a href="<?php echo RUTA_URL ?>/public/grupos/editar/<?php echo $grupo[0] ?>" class="button radius tiny secondary
                <button name="eliminar" value="<?php echo $grupo[0] ?>" class="button radius tiny secondary">Eliminar</button>
              </td>
            </tr>
          <?php endforeach ?>
      </tbody>
    </table>
  <?php else: ?>
    <p>No hay registros</p> <!-- En caso de que no haya ningún registro en la base de datos se mostrará este mensaje -->
  <?php endif ?>
```

Registro de un grupo

Una vez que el formulario se llena, éste en enviado a una función llamada *agregar()* ubicada en el controlador grupos.

```
//Página agregar grupo
public function agregar(){
  //Es true cuando un post se envia, en este caso cuando se envía el formulario de registro
  if($_SERVER["REQUEST_METHOD"] == "POST"){
    $datos = [
      "nombre" => trim($_POST["nombre"]),
    ];

    $this->modelo->agregarGrupo($datos); //Se agrega el grupo

    $this->index(); //Se regresa al index
  }
  else{
    $this->view("grupos/agregar");
  }
}
```

La función guarda los datos en un arreglo y lo envía a la función *agregarGrupo($datos)* ubicada en el modelo grupo.

```
//Método para agregar un grupo
public function agregarGrupo($data){
  $this->trans->query("INSERT INTO grupos VALUES(null, :nombre)");

  $this->trans->bind(":nombre", $data["nombre"]);

  $res = $this->trans->execute();
}
```

Esta función se encarga de guardar los datos en la base de datos.

Una vez que se haya guardado el grupo en la base de datos, el controlador grupos redireccionará hacia la página donde está la tabla de grupos.

Modificación de un grupo

Una vez que el formulario se llena, éste en enviado a una función llamada edita*r()* ubicada en el controlador grupos.

```
//Página editar grupo
public function editar($id = ""){
  //Es true cuando un post se envia, en este caso cuando se envía el formulario
  if($_SERVER["REQUEST_METHOD"] == "POST"){
    $datos = [
      "id" => trim($_POST["id"]),
      "nombre" => trim($_POST["nombre"]),
    ];

    $this->modelo->editarGrupo($datos); //Se edita el grupo

    $this->index(); //Se regresa al index
  }
  else{
    $this->view("grupos/editar", $this->modelo->getGrupo($id));
  }
}
```

La función guarda los datos en un arreglo y lo envía a la función *editarGrupo($datos)* ubicada en el modelo grupos.

```
//Método para editar un grupo
public function editarGrupo($data){
  $this->trans->query("UPDATE grupos SET nombre=:nombre WHERE id=:id");

  $this->trans->bind(":id", $data["id"]);
  $this->trans->bind(":nombre", $data["nombre"]);

  $res = $this->trans->execute();
}
```

Esta función se encarga de actualizar los datos del grupo en la base de datos.

Una vez que se haya guardado el grupo en la base de datos, el controlador grupos redireccionará hacia la página donde está la tabla de grupos.

Eliminación de un grupo

Al hacer clic en el botón para eliminar, se hace una llamada a la función *eliminar()* ubicada en el controlador de grupos. Ésta hace una llamada a la función *eliminarGrupo()* ubicada en el modelo de grupos, y le envía el id del grupo a eliminar.

```
//Método para eliminar grupo
public function eliminar($id = ""){

  $this->modelo->eliminarGrupo($id); //Se elimina al grupo

  $this->index(); //Se regresa al index
}
```

Dentro de la función del modelo, se hace una consulta en la base de datos, la cual elimina primero a las alumnas que pertenezcan a ese grupo, y luego al grupo que tenga el id pasado a la función. Después de eso, el controlador redireccionará a la página de inicio de grupos.

```
//Método para eliminar un grupo
public function eliminarGrupo($id){
  //Primero se eliminan las alumnas del grupo
  $this->trans->query("DELETE FROM alumnas WHERE grupo=:id");
  $this->trans->bind(":id", $id);
  $res = $this->trans->execute();

  //Se elimina el grupo
  $this->trans->query("DELETE FROM grupos WHERE id=:id");
  $this->trans->bind(":id", $id);
  $res = $this->trans->execute();
}
```

Visualización de pagos (tabla de pagos)

El controlador home se encarga de direccionar el sitio hacia la vista en donde se encuentra la tabla (la página pagos), y le envía como parámetros un arreglo con los datos de los pagos, los cuales se obtienen mediante otra función ubicada en el modelo home.

```
//Método para ir a la página para visualizar los lugares de registros
public function pagos(){
  $this->view("home/pagos", $this->modelo->getPagos());
}
```

La función del modelo se encarga de hacer una consulta a la base de datos en la que se obtienen los datos de los pagos. Una vez obtenidos, retorna un arreglo con los pagos.

```
//Método para obtener todos los pagos
public function getPagos(){
  $this->trans->query("SELECT p.id, g.nombre, CONCAT(a.nombre, ' ', a.apellido), CONCAT(p.nombre_mama, ' ', p.apellido_mama), p.fecha_p
  $res = $this->trans->execute();

  return $res->fetchAll();
}
```

Después, en la vista de pagos se obtiene la información del arreglo pasado y mediante un ciclo se muestran los datos en forma de tabla.

```
<!-- Se verifica que el arreglo de alumnos no esté vacío -->
<?php if(!empty($data)):?>
    <table>
      <thead>
        <tr>
          <!-- Datos de los alumnos -->
          <th>Lugar</th>
          <th>Grupo</th>
          <th>Nombre alumna</th>
          <th>Nombre mama</th>
          <th>Fecha de pago</th>
          <th>Fecha de envío</th>
          <th>Folio</th>
          <th>Comprobante de folio</th>
        </tr>
      </thead>
      <tbody>
        <!-- Ciclo para imprimir todas las lugares y su respectiva información -->
        <?php foreach($data as $pago): ?>
            <tr>
              <td><?php echo $pago[0] ?></td>
              <td><?php echo $pago[1] ?></td>
              <td><?php echo $pago[2] ?></td>
              <td><?php echo $pago[3] ?></td>
              <td><?php echo $pago[4] ?></td>
              <td><?php echo $pago[5] ?></td>
              <td><?php echo $pago[6] ?></td>
              <td><a href="<?php echo RUTA_URL . substr($pago[7], 56) ?>">Ver</a> </td>
            </tr>
          <?php endforeach ?>
      </tbody>
    </table>
  <?php else: ?>
    <p>No hay registros</p> <!-- En caso de que no haya ningún registro en la base de datos se mostrará este mensaje -->
  <?php endif ?>
```