Universidad Politécnica de Victoria

Ingeniería en Tecnologías de la Información

Manual Técnico – Sistema Danzlife

Examen Unidad 2

Alumno: **Erick Elizondo Rodríguez**

Matricula: **1530061**

Materia: **Tecnologías y Aplicaciones Web**

Profesor: **M.S.I. Mario Humberto Rodríguez Chávez**

El sistema de Danzlife está encargado de controlar el ingreso de los ticket de pago de unas alumnas, ofreciendo una interfaz publica para este procedimiento, además ofreciendo una parte donde se ofrecen herramientas para un administrador de la página, que controla los valores de la interfaz publica, controlando las alumnas, los grupos a las que estas pertenecen y además los pagos que estas realizan.

Ofreciéndose además un control para usuarios administradores, donde se pueden agregar cuantos usuarios sean requeridos.

Funciones:

Controller.php y Crud.php

1. enlacesPaginasController()

```php
public function enlacesPaginasController(){
    if(isset( $_GET['action']))
        $enlaces = $_GET['action'];
    else
        $enlaces = "index";
    $respuesta = Paginas::enlacesPaginasModel($enlaces);
    include $respuesta;
    return $respuesta;
}
```

Solicita a la página que se esté solicitando dependiendo de la variable get de action encontrada en la url solicitada y en caso de que no se encuentre nada manda al index.php.

2. ingresoUsuarioController()

```php
public function ingresoUsuarioController(){

    if(isset($_POST["user"])){
        $datosController = array( "user"=>$_POST["user"],
                                  "pass"=>$_POST["pass"]);
        $respuesta = Datos::ingresoUsuarioModel($datosController, "users");

        if($respuesta["user"] == $_POST["user"] && $respuesta["pass"] == md5($_POST["pass"])){
            if(!isset($_SESSION))
                session_start();
            $_SESSION["validar"] = true;
            header("location:index.php?action=dashboard");
        }
        else{
            header("location:index.php?action=fallo");
        }
    }
}
```

Mediante una solicitud al modelo se obtienen la contraseña del usuario que coincida con el usuario ingresado, donde posteriormente se verifica que estos coincidan, además aplicando una encriptación md5 a la contraseña, pera después iniciar la sesión y re direccionar al dashboard.

3. upload()

```php
public function upload(){
    $target_dir = dirname($_SERVER["SCRIPT_FILENAME"])."/views/images/";
    $new_name= date("Y-m-d-H-i-s") . basename($_FILES["fileToUpload"]["name"]);
    $target_file = $target_dir . $new_name;
    $uploadOk = 1;
    $imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
    $error="";

    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false)
        $uploadOk = 1;
    else {
        $error = "El archivo no es una imagen.";
        $uploadOk = 0;
    }
    if (file_exists($target_file)) {
        $error = "El archivo ya existe.";
        $uploadOk = 0;
    }
    if ($_FILES["fileToUpload"]["size"] > 5000000) {
        $error = "El archivo es muy grande.";
        $uploadOk = 0;
    }
    if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "PNG" && $imageFileType != "JPG" && $imageFileType != "jpeg") {
        $error = "Solo puede subir archivos jpg y png";
        $uploadOk = 0;
    }
    if ($uploadOk == 0) {
        $error = "El archivo no pudo ser subido.";
    } else {
        if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
            echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
        } else {
            $error = "No se pudo subir el archivo";
            echo('<script> swal("Error", "'.$error.'", "error");</script>');
        }
    }
    return $new_name;
}
```

Esta función permite el control de la subida de archivos de imágenes, que ese encuentren en la variable global del servidor $_FILES este código al ser llamado, obtiene el archivo cargado y lo manda a la carpeta temporal del servidor y posteriormente este es verificado, para que se cheque si cumple con todos los parámetros necesarios, como que tenga extensiones de archivo permitidas o un tamaño adecuado, para posteriormente terminar la carga al servidor.

4. getSelect()

```php
public function getSelect(){
    $respuesta_grupos = Datos::obtenerGruposModel("grupos");

    $st_grupo="";
    for($i=0;$i<sizeof($respuesta_grupos);$i++)
        $st_grupo=$st_grupo."<option value='".$respuesta_grupos[$i]['id']."'>".$respuesta_grupos[$i]['nombre']."</option>";

    return $st_grupo;
}
```

Genera una cadena de texto que contiene las opciones para alimentar un select correspondiente a los grupos registrados en el sistema, generando en si el código html con sus valores y nombres respectivamente.

5.getAlumnasFromId()

```php
public function getAlumnasFromId($id){
    $respuesta_grupos = Datos::obtenerAlumnasFromGroupModel($id,"alumnas");

    $st_grupo="";
    for($i=0;$i<sizeof($respuesta_grupos);$i++)
        $st_grupo=$st_grupo."<option value='".$respuesta_grupos[$i]['id']."'>".$respuesta_grupos[$i]['nombre']."</option>";

    return $st_grupo;
}
```

Genera una cadena de texto que contiene el código HTML para llenar las opciones un select de alumnas correspondiente a un id de grupo mandado por parámetro.

6. vistaAlumnaController()

```php
public function vistaAlumnaController(){

    $respuesta = Datos::vistaAlumnasModel("alumnas");

    foreach($respuesta as $row => $item){
    echo'<tr>
            <td>'.$item["nombre"].'</td>
            <td>'.$item["apellido_paterno"].'</td>
            <td>'.$item["grupo"].'</td>
            <td><a href="index.php?action=editar_alumna&id='.$item["id"].'"><button class="btn btn-block btn-warning btn-md">Editar</button></a></td>
            <td><button onClick="wait();" class="btn btn-block btn btn-danger btn-md">Borrar</button></a></td>
        </tr>';
    echo '
        <script type="text/javascript">
            function wait(){
                swal({
                  title: "¿Esta seguro que desea eliminar esta alumna?",
                  text: "Al eliminar esta alumna tambien se eliminara toda la informacion relacionada a la misma",
                  icon: "warning",
                  buttons: true,
                  dangerMode: true,
                })
                .then((willDelete) => {
                  if (willDelete) {
                    window.location.href = "./index.php?action=alumnas&idBorrar='.$item["id"].'";
                  }
                });
            }
        </script>';
    }
}
```

Genera la interfaz del crud de los alumnos, donde se ofrece una tabla con todas las alumnas, además agregando unos botones que permiten eliminar, modificar y también el agregar un nuevo registro de alumnas.

7. registroBaseAlumnaController()

```php
public function registroBaseAlumnaController(){
    $respuesta_grupos = Datos::obtenerGruposModel("grupos");

    $st_grupo="";
    for($i=0;$i<sizeof($respuesta_grupos);$i++)
        $st_grupo=$st_grupo."<option value='".$respuesta_grupos[$i]['id']."'>".$respuesta_grupos[$i]['nombre']."</option>";

    echo'<label for="nombre">Nombre:</label>
        <input class="form-control" type="text" name="nombre" required>
        <label for="apellido_paterno">Apellido Paterno:</label>
        <input class="form-control" type="text" name="apellido_paterno" required>
        <label for="id_grupo">Grupo:</label>
        <select id="categorias" class="form-control js-example-basic-multiple" name="id_grupo">
            '.$st_grupo.'
        </select>
        <br>
        <div class="card-footer">
           <input type="submit" class="btn btn-primary float-right" value="Registrar">
        </div>';
}
```

Genera la interfaz base para el registro de algún alumno, preparando todo el código HTML y JavaScript para el llenado de un select.

8. registroAlumnaController()

```php
public function registroAlumnaController(){

    if(isset($_POST["nombre"])){
        $datosController = array(
                                    "nombre"=>$_POST["nombre"],
                                    "apellido_paterno"=>$_POST["apellido_paterno"],
                                    "id_grupo"=>$_POST["id_grupo"]
                                );

        $respuesta = Datos::registroAlumnaModel($datosController, "alumnas");


        if($respuesta == "success"){
            header("location:index.php?action=ok_alumna");
        }
        else{
            header("location:index.php");
        }

    }

}
```

Permite el agregado de una alumna a la base de datos, que llena un array que será utilizado posteriormente en el modelo llenado a través del método post de los componentes de la interfaz alumnas, posteriormente re direcciona al crud de alumnas en caso de hacer un registro exitoso.

9. borrarAlumnaController()

```php
public function borrarAlumnaController(){

    if(isset($_GET["idBorrar"])){
        $datosController = $_GET["idBorrar"];

        $respuesta = Datos::borrarAlumnaModel($datosController, "alumnas");

        if($respuesta == "success"){
            header("location:index.php?action=alumnas");
        }else{
            echo('<script> swal("Error", "La alumna no pudo ser eliminada", "error");</script>');
        }
    }
}
```

Se encarga de la eliminación de una alumna dependiendo de un id mandado por método get en la url del archivo.

10. editarAlumnaController()

```php
public function editarAlumnaController(){
    $datosController = $_GET["id"];
    $respuesta = Datos::editarAlumnaModel($datosController, "alumnas");

    $respuesta_grupos = Datos::obtenerGruposModel("grupos");

    $st_grupo="";
    for($i=0;$i<sizeof($respuesta_grupos);$i++)
        $st_grupo=$st_grupo."<option value='".$respuesta_grupos[$i]['id']."'>".$respuesta_grupos[$i]['nombre']."</option>";

    echo'<input type="hidden" value="'.$respuesta['id'].'" name="id">
         <label for="nombre">Nombre:</label>
         <input class="form-control" type="text" name="nombre" value="'.$respuesta["nombre"].'" required>
         <label for="apellido_paterno">Apellido Paterno:</label>
         <input class="form-control" type="text" name="apellido_paterno" value="'.$respuesta["apellido_paterno"].'" required>
         <label for="id_grupo">Grupo:</label>
         <select id="categorias" class="form-control js-example-basic-multiple" name="id_grupo">
              '.$st_grupo.'
         </select>
         <br>
         <div class="card-footer">
            <input type="button" onclick="wait();" class="btn btn-primary float-right" value="Actualizar">
         </div>';

    echo'
    <script>

        function wait(){
            swal({
              title: "¿Esta seguro que desea modificar esta alumna?",
              icon: "warning",
              buttons: true,
              dangerMode: true,
            })
            .then((willDelete) => {
              if (willDelete) {
                document.getElementById("cat").submit();
              }
            });
        }
    </script>';
```

Permite la edición de un alumno, generando la vista mediante un echo, que contiene todos los componentes que conformaran la misma, además, teniendo un select llenado con todos los grupos registrados en el sistema y además una función de

JavaScript llamada wait() la cual genera un mensaje de confinación para la eliminación de la alumna.

11. actualizarAlumnaController()

```php
public function actualizarAlumnaController(){
    if(isset($_POST["id"])){
        $datosController = array( "id"=>$_POST["id"],
                                  "nombre"=>$_POST["nombre"],
                                  "apellido_paterno"=>$_POST["apellido_paterno"],
                                  "id_grupo"=>$_POST["id_grupo"],
                                );
        $respuesta = Datos::actualizarAlumnaModel($datosController, "alumnas");

        if($respuesta == "success"){
            header("location:index.php?action=cambio_alumna");
        }
        else{
            echo "error";
        }
    }
}
```

Permite actualizar el registro de la base de datos correspondiente a una alumna con el id correspondiente recibido mediante el método post, ingresando todos los valores que se encuentren en los componentes de la página anterior y que de igual manera son mandados al hacer un submit.

12. vistaGrupoController()

```php
public function vistaGrupoController(){

    $respuesta = Datos::vistaGrupoModel("grupos");  #Cargando todos los grupos

    #Llenando la tabla
    foreach($respuesta as $row => $item){
    echo'<tr>
            <td>'.$item["id"].'</td>
            <td>'.$item["nombre"].'</td>
            <td><a href="index.php?action=editar_grupo&id='.$item["id"].'"><button class="btn btn-block btn-warning btn-md">Editar</button></a></td>
            <td><button onClick="wait();" class="btn btn-block btn btn-danger btn-md">Borrar</button></a></td>
        </tr>';
    echo '
        <script type="text/javascript">
            function wait(){
                swal({
                  title: "¿Esta seguro que desea eliminar este grupo?",
                  text: "Al eliminar este grupo tambien se eliminara toda la informacion relacionada a la misma",
                  icon: "warning",
                  buttons: true,
                  dangerMode: true,
                })
                .then((willDelete) => {
                  if (willDelete) {
                    window.location.href = "./index.php?action=grupos&idBorrar='.$item["id"].'";
                  }
                });
            }
        </script>';
    }
}
```

Permite generar la vista del grupo, generando una tabla que contiene todos los registros de la base de datos y unos botones para la eliminación y modificación en el cual se redirige a la vista de modificación.

13. registroBaseGrupoController()

```php
public function registroBaseGrupoController(){
    $respuesta_grupos = Datos::obtenerGruposModel("grupos");

    echo'<label for="nombre">Nombre:</label>
        <input class="form-control" type="text" name="nombre" required>
        <br>
        <div class="card-footer">
           <input type="submit" class="btn btn-primary float-right" value="Registrar">
        </div>';
}
```

Genera la interfaz base para el registro de un grupo, agregando los componentes que conforman la interfaz requerida para su modificación.

14. registroGrupoController()

```php
public function registroGrupoController(){

    if(isset($_POST["nombre"])){
        $datosController = array("nombre"=>$_POST["nombre"]);

        $respuesta = Datos::registroGrupoModel($datosController, "grupos");

        if($respuesta == "success")
            header("location:index.php?action=ok_grupo");
        else
            header("location:index.php");
    }

}
```

Controlador que permite obtener el nombre por post al registrar un grupo en el sistema, redirigiendo a la interfaz de grupos en caso de que esta sea exitosa.

15. borrarGrupoController()

```php
public function borrarGrupoController(){

    if(isset($_GET["idBorrar"])){
        $datosController = $_GET["idBorrar"];

        $respuesta = Datos::borrarAlumnaModel($datosController, "grupos");

        if($respuesta == "success")
            header("location:index.php?action=grupos");
        else
            echo('<script> swal("Error", "El grupo no pudo ser eliminada", "error");</script>');
    }
}
```

Controlador que permite realizar el borrado de algún grupo dependiendo del id que se le mande, este recibido mediante el método get en php, utilizando el modelo borrarAlumnaModel y redirigiendo a todos los grupos en caso de lograr hacer el borrado del sistema y en caso contrario mostrando un sweetAlert con el error.

16. editarGrupoController()

```php
public function editarGrupoController(){
    $datosController = $_GET["id"];
    $respuesta = Datos::editarGrupoModel($datosController, "grupos");

    echo'<input type="hidden" value="'.$respuesta['id'].'" name="id">
        <label for="nombre">Nombre:</label>
        <input class="form-control" type="text" name="nombre" value="'.$respuesta["nombre"].'" required>
        <br>
        <div class="card-footer">
           <input type="button" onclick="wait();" class="btn btn-primary float-right" value="Actualizar">
        </div>';

    echo'
    <script>

        function wait(){
            swal({
              title: "¿Esta seguro que desea modificar este grupo?",
              icon: "warning",
              buttons: true,
              dangerMode: true,
            })
            .then((willDelete) => {
              if (willDelete) {
                document.getElementById("cat").submit();
              }
            });
        }
    </script>';
}
```

Este controlador permite editar un grupo generando la interfaz con los valores actuales de un registro dependiendo de su id y llenando todos los componentes con los valores retribuidos del registro actual, además mostrando una confirmación para realizar la modificación.

17. actualizarGrupoController()

```php
public function actualizarGrupoController(){
    if(isset($_POST["id"])){
        $datosController = array( "id"=>$_POST["id"],
                                  "nombre"=>$_POST["nombre"]
                                );
        $respuesta = Datos::actualizarGrupoModel($datosController, "grupos");

        if($respuesta == "success"){
            header("location:index.php?action=cambio_grupo");
        }
        else{
            echo "error";
        }
    }
}
```

Permite la actualización de un registro de grupo en la base de datos, dependiendo del id que se mande mediante el método post, recibiendo su id y nombre y mandándolo al modelo actualizarGrupoModel para después obtener la respuesta que se obtiene del mismo y determinado si existió una actualización exitosa.

18. registroPagoController ()

```php
public function registroPagoController(){
    if(isset($_POST["id_alumna"])){
        $file="";
        if(isset($_FILES))
            $file=$this->upload($_FILES);
        $datosController = array(
                                "id_alumna"=>$_POST["id_alumna"],
                                "nombre_mama"=>$_POST["nombre_mama"]." ".$_POST["apellido_mama"],
                                "fecha_pago"=>date('Y-m-d H:i:s', strtotime($_POST["fecha_pago"])),
                                "fecha_envio"=>date("Y-m-d H:i:s"),
                                "url_imagen"=>$file,
                                "folio"=>$_POST["folio"]
                            );

        $respuesta = Datos::registroPagoModel($datosController, "pagos");

        if($respuesta == "success"){
            header("location:index.php?action=p_lugares");
        }
        else{
            header("location:index.php");
        }
    }

}
#Vista de los pagos
```

Permite realizar un registro de pago donde se reciben todos los atributos del registro mediante el método post y el archivo que se subirá mediante el método files para después llamar la función upload la cual lo subirá al servidor y posteriormente

realizar el registro mediante el modelo registroPagoModel para enviarlo a la base de datos.

19. vistaPublicPagoController()

```php
public function vistaPublicPagoController(){

    $respuesta = Datos::vistaPublicPagoModel("pagos");

    #Llenando la tabla
    foreach($respuesta as $row => $item){
    echo'<tr>
            <td>'.$item["id"].'</td>
            <td>'.$item["nombre"].'</td>
            <td>'.$item["nombre_mama"].'</td>
            <td>'.$item["fecha_envio"].'</td>
            <td>'.$item["folio"].'</td>
        </tr>';
    }
}
```

Permite el llenado de una tabla, imprimiendo todas las filas con los atributos de los registros de los pagos.

20. vistaPagoController()

```php
public function vistaPagoController(){

    $respuesta = Datos::vistaPagoModel("pagos");    #Cargando todos los grupos

    #Llenando la tabla
    foreach($respuesta as $row => $item){
        echo'<tr>
                <td>'.$item["id"].'</td>
                <td>'.$item["nombre"].'</td>
                <td>'.$item["nombre_mama"].'</td>
                <td>'.$item["fecha_envio"].'</td>
                <td>'.$item["fecha_pago"].'</td>
                <td>'.$item["folio"].'</td>
                <td><a href="./views/images/'.$item["url_imagen"].'"><button class="btn btn-block btn-info btn-md">Ver Imagen</button></a></td>
                <td><a href="index.php?action=editar_pago&id='.$item["id"].'"><button class="btn btn-block btn-warning btn-md">Editar</button></a></td>
                <td><button onClick="wait('.$item['id'].');" class="btn btn-block btn btn-danger btn-md">Borrar</button></a></td>
            </tr>';

    echo '
        <script type="text/javascript">
            function wait(id){
                swal({
                  title: "¿Esta seguro que desea eliminar este pago?",
                  text: "Al eliminar esta pago tambien se eliminara toda la informacion relacionada a la misma",
                  icon: "warning",
                  buttons: true,
                  dangerMode: true,
                })
                .then((willDelete) => {
                  if (willDelete) {
                    window.location.href = "./index.php?action=pagos&idBorrar="+id;
                  }
                });
            }
        </script>';
    }
}
```

Permite la generación de la vista de los pagos, llenando una tabla donde se muestran todos los registros de los pagos que se encuentran en el sistema, además

agregando una serie de botones los cuales son ver_imagen, modificar y eliminar, donde al ver imagen se redirige a la ubicación real de la imagen en el servidor y en modificar que re direcciona a otra página del sistema para realizar la modificación y el eliminar la cual es llamada mediante un método donde se solicita una confirmación para continuar con la eliminación.

21. borrarPagoController()

```php
public function borrarPagoController(){

    if(isset($_GET["idBorrar"])){
        $datosController = $_GET["idBorrar"];

        $respuesta = Datos::borrarPagoModel($datosController, "pagos");

        if($respuesta == "success")
            header("location:index.php?action=pagos");
        else
            echo('<script> swal("Error", "El grupo no pudo ser eliminada", "error");</script>');
    }
}
```

Permite eliminar registros de pagos que se encuentren registrados en el sistema, esto dependiendo de un id que se mande a través de la url de la vista.

22. editarPagoController()

```php
public function editarPagoController(){
    $datosController = $_GET["id"];
    $respuesta = Datos::editarPagosModel($datosController, "pagos");

    $st_grupo = $this->getSelect();


    echo'<label for="id_grupo">Seleccione grupo</label>
        <input type="hidden" name="id_pago" value="'.$respuesta['id_pago'].'"></input>
        <select id="grupos" class="form-control js-example-basic-multiple" name="id_grupo" required>
          '.$st_grupo.'?>
        </select>
        <br>
        <label for="id_grupo">Seleccione alumna</label>
        <select id="alumnas" class="form-control js-example-basic-multiple" name="id_alumna" required>
        </select>
        <br>
        <div class="row">
            <div class="col-md-6">
                <label for="nombre_mama">Nombre de la mama</label>
                <input class = "form-control" type="text" name="nombre_mama" value="'.$respuesta['nombre_mama'].'" required>
            </div>
        </div>
```

Permite la edición de un pago con respecto al id que recibe mediante el método post para después llenar la interfaz con los valores actuales del registro y poder realizar su modificación.

23. actualizarPagoController()

```php
public function actualizarPagoController(){
    if(isset($_POST["id_pago"])){
        $datosController = array(
                                "id_pago"=>$_POST["id_pago"],
                                "id_alumna"=>$_POST["id_alumna"],
                                "nombre_mama"=>$_POST["nombre_mama"],
                                "fecha_pago"=>$_POST["fecha_pago"],
                                "fecha_envio"=>$_POST["fecha_envio"],
                                "folio"=>$_POST["folio"]
                               );
        $respuesta = Datos::actualizarPagoModel($datosController, "pagos");

        if($respuesta == "success"){
            header("location:index.php?action=cambio_pago");
        }
        else{
            echo "error";
        }
    }
}
```

Permite el update de un pago, recibiendo mediante post todos los valores del formulario para después mandarlos al modelo y realizar la actualización.

24. ingresoUsuarioModel()

```php
public function ingresoUsuarioModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("SELECT user, pass FROM $tabla WHERE user = :user");
    $stmt->bindParam(":user", $datosModel["user"], PDO::PARAM_STR);
    $stmt->execute();

    return $stmt->fetch();
    $stmt->close();
}
```

Función que retorna el usuario y contraseña de la tabla users donde el usuario coincida a un usuario mandado.

25. obtenerGruposModel()

```php
public function obtenerGruposModel($tabla){
    $stmt = Conexion::conectar()->prepare("SELECT id, nombre FROM $tabla");
    $stmt->execute();

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

Función que permite obtener todos los valores de los grupos registrados en la base de datos, donde se obtiene su id y su nombre.

26. obtenerAlumnasFromGroupModel()

```php
public function obtenerAlumnasFromGroupModel($id,$tabla){
    $stmt = Conexion::conectar()->prepare("SELECT id, concat(nombre, ' ',apellido_paterno) as nombre FROM $tabla WHERE id_grupo=:id_grupo");
    $stmt->bindParam(":id_grupo", $id, PDO::PARAM_INT);
    $stmt->execute();

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

Esta función permite obtener el id y el nombre completo de las alumnas que pertenezcan a determinado grupo.

27. borrarAlumnaModel()

```php
public function borrarAlumnaModel($datosModel, $tabla){
    $stmt = Conexion::conectar()->prepare("DELETE FROM $tabla WHERE id = :id");
    $stmt->bindParam(":id", $datosModel, PDO::PARAM_INT);

    if($stmt->execute())
        return "success";
    else
        return "error";

    $stmt->close();

}
```

Función que se encarga del eliminado de una alumna de la base de datos.

28. registroAlumnaModel()

```php
public function registroAlumnaModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("INSERT INTO $tabla (nombre, apellido_paterno, id_grupo) VALUES (:nombre, :apellido_paterno, :id_grupo)");

    $stmt->bindParam(":nombre", $datosModel["nombre"], PDO::PARAM_STR);
    $stmt->bindParam(":apellido_paterno", $datosModel["apellido_paterno"], PDO::PARAM_STR);
    $stmt->bindParam(":id_grupo", $datosModel["id_grupo"], PDO::PARAM_INT);

    if($stmt->execute()){
        return "success";
    }
    else{
        return "error";
    }

    $stmt->close();
}
```

Función que se encarga del registro de una alumna en la base de datos, ingresando su nombre, apellido paterno y el id del grupo a que esta pertenece.

29. vistaAlumnasModel()

```php
public function vistaAlumnasModel($tabla){

    $stmt = Conexion::conectar()->prepare("SELECT a.id as id, a.nombre as nombre, a.apellido_paterno as apellido_paterno, g.nombre as grupo from $tabla AS a
        INNER JOIN grupos AS g on g.id=a.id_grupo");
    $stmt->execute();

    return $stmt->fetchAll();

    $stmt->close();
}
```

Obtiene todos los valores para el llenado de la vista principal donde se muestran todas las alumnas, agregando un inner join con los grupos para mostrar el número de grupo y no el id registrado.

30. editarAlumnaModel()

```php
public function editarAlumnaModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("SELECT id, nombre, apellido_paterno, id_grupo FROM $tabla WHERE id = :id");
    $stmt->bindParam(":id", $datosModel, PDO::PARAM_INT);

    $stmt->execute();

    return $stmt->fetch();

    $stmt->close();

}
```

Permite la edición de una alumna que es identificada mediante su id, obteniendo todos los atributos actualmente registrados de la misma.

31. actualizarAlumnaModel()

```php
public function actualizarAlumnaModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre, apellido_paterno = :apellido_paterno WHERE id_grupo = :id_grupo");

    $stmt->bindParam(":nombre", $datosModel['nombre'], PDO::PARAM_STR);
    $stmt->bindParam(":apellido_paterno", $datosModel["apellido_paterno"], PDO::PARAM_INT);
    $stmt->bindParam(":id_grupo", $datosModel["id_grupo"], PDO::PARAM_INT);

    if($stmt->execute())
        return "success";
    else
        return "error";

    $stmt->close();
}
```

Realiza el update de una alumna específica en la base de datos reemplazando los valores actuales por unos mandados mediante un array llamado datosModel.

32. borrarGrupoModel()

```php
public function borrarGrupoModel($datosModel, $tabla){
    $stmt = Conexion::conectar()->prepare("DELETE FROM $tabla WHERE id = :id");
    $stmt->bindParam(":id", $datosModel, PDO::PARAM_INT);

    if($stmt->execute())
        return "success";
    else
        return "error";

    $stmt->close();
}
```

Permite la eliminación de un grupo específico dependiendo del id que se mande mediante parámetro.

33. registroGrupoModel()

```php
public function registroGrupoModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("INSERT INTO $tabla (nombre) VALUES (:nombre)");

    $stmt->bindParam(":nombre", $datosModel["nombre"], PDO::PARAM_STR);

    if($stmt->execute())
        return "success";
    else
        return "error";

    $stmt->close();
}
```

Permite el registro de un grupo en la base de datos, ingresando solamente el nombre del grupo y retornando una respuesta success en caso de éxito o error en caso de fallo.

34. vistaGrupoModel()

```php
public function vistaGrupoModel($tabla){

    $stmt = Conexion::conectar()->prepare("SELECT id, nombre FROM $tabla");
    $stmt->execute();

    return $stmt->fetchAll();

    $stmt->close();
}
```

Permite obtener todos los valores de la tabla de grupos para la vista principal donde se muestran todos los registros a manera de tabla.

35. editarGrupoModel()

```php
public function editarGrupoModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("SELECT id, nombre FROM $tabla WHERE id = :id");
    $stmt->bindParam(":id", $datosModel, PDO::PARAM_INT);

    $stmt->execute();

    return $stmt->fetch();

    $stmt->close();

}
```

Permite la edición de un grupo dependiendo del id que se mande, obteniendo los valores del registro actual.

36. actualizarGrupoModel()

```php
public function actualizarGrupoModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre WHERE id = :id");
    $stmt->bindParam(":id", $datosModel['id'], PDO::PARAM_INT);
    $stmt->bindParam(":nombre", $datosModel['nombre'], PDO::PARAM_STR);

    if($stmt->execute())
        return "success";
    else
        return "error";

    $stmt->close();
}
```

Permite el update de un grupo, actualizando su nombre con respecto a algún id que se mande.

37. borrarPagoModel()

```php
public function borrarPagoModel($datosModel, $tabla){
    $stmt = Conexion::conectar()->prepare("DELETE FROM $tabla WHERE id_pago = :id_pago");
    $stmt->bindParam(":id_pago", $datosModel, PDO::PARAM_INT);

    if($stmt->execute())
        return "success";
    else
        return "error";

    $stmt->close();
}
```

Se encarga del eliminado de un pago, donde dependiendo del id que se mande se borra todo y todos los registros que estén relacionados con este pago.

38. editarPagosModel()

```php
public function editarPagosModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("SELECT a.id_grupo as id_grupo, p.id_pago as id_pago, p.id_alumna as id_alumna, p.nombre_mama as nombre_mama, p.
        fecha_pago as fecha_pago, p.fecha_envio as fecha_envio, p.url_imagen as url_imagen, p.folio as folio FROM $tabla as p INNER JOIN alumnas as A on a.id=p.
        id_alumna WHERE p.id_pago = :id_pago");
    $stmt->bindParam(":id_pago", $datosModel, PDO::PARAM_INT);

    $stmt->execute();

    return $stmt->fetch();

    $stmt->close();

}
```

Se encarga de controlar la edición de un pago al obtener primeramente los valores para el llenado de la interfaz con los mismos.

39. actualizarPagoModel()

```php
public function actualizarPagoModel($datosModel, $tabla){
    var_dump($datosModel);

    $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET id_alumna = :id_alumna, nombre_mama = :nombre_mama, fecha_pago =:fecha_pago, fecha_envio =
        :fecha_envio, folio=:folio WHERE id_pago = :id_pago");
    +
    $stmt->bindParam(":id_pago", $datosModel['id_pago'], PDO::PARAM_INT);
    $stmt->bindParam(":id_alumna", $datosModel['id_alumna'], PDO::PARAM_INT);
    $stmt->bindParam(":nombre_mama", $datosModel['nombre_mama'], PDO::PARAM_STR);
    $stmt->bindParam(":fecha_pago", $datosModel['fecha_pago'], PDO::PARAM_STR);
    $stmt->bindParam(":fecha_envio", $datosModel['fecha_envio'], PDO::PARAM_STR);
    $stmt->bindParam(":folio", $datosModel['folio'], PDO::PARAM_STR);

    $stmt->errorInfo();
    if($stmt->execute())
        return "success";
    else{
        return "error";
    }

    $stmt->close();
}
```

Se encarga de realizar el update en los pagos, recibiendo todos los valores de los atributos que tiene esta tabla.

40. registroPagoModel()

```php
public function registroPagoModel($datosModel, $tabla){

    $stmt = Conexion::conectar()->prepare("INSERT INTO $tabla (id_alumna, nombre_mama, fecha_pago, fecha_envio, url_imagen, folio) VALUES (:id_alumna,
        :nombre_mama, :fecha_pago, :fecha_envio, :url_imagen, :folio)");

    $stmt->bindParam(":id_alumna", $datosModel["id_alumna"], PDO::PARAM_INT);
    $stmt->bindParam(":nombre_mama", $datosModel["nombre_mama"], PDO::PARAM_STR);
    $stmt->bindParam(":fecha_pago", $datosModel["fecha_pago"], PDO::PARAM_STR);
    $stmt->bindParam(":fecha_envio", $datosModel["fecha_envio"], PDO::PARAM_STR);
    $stmt->bindParam(":url_imagen", $datosModel["url_imagen"], PDO::PARAM_STR);
    $stmt->bindParam(":folio", $datosModel["folio"], PDO::PARAM_STR);

    if($stmt->execute())
        return "success";
    else
        return "error";

    $stmt->close();
}
```

Se controla el registro de un pago donde se ingresan todos los valores de un pago a la base de datos para crear un nuevo registro.

41. vistaPublicPagoModel()

```php
public function vistaPublicPagoModel($tabla){

    $stmt = Conexion::conectar()->prepare("SELECT p.id_pago as id, CONCAT(a.nombre,' ',a.apellido_paterno) as nombre, p.nombre_mama AS nombre_mama, p.fecha_envio
        as fecha_envio, p.folio AS folio FROM $tabla AS p INNER JOIN alumnas AS a ON a.id=p.id_alumna");
    $stmt->execute();

    return $stmt->fetchAll();

    $stmt->close();
}
```

Permite obtener los atributos de alguna alumna que se mande a través de su id, además obteniendo el nombre de las alumnas, para mostrarlo en la interfaz publica del sistema donde es necesario excluir ciertos atributos.

42.vistaPagoModel()

```php
public function vistaPagoModel($tabla){

    $stmt = Conexion::conectar()->prepare("SELECT p.id_pago as id, CONCAT(a.nombre,' ',a.apellido_paterno) as nombre, p.nombre_mama AS nombre_mama, p.fecha_envio
        as fecha_envio, p.fecha_pago as fecha_pago, p.url_imagen as url_imagen, p.folio AS folio FROM $tabla AS p INNER JOIN alumnas AS a ON a.id=p.id_alumna");


    $stmt->execute();

    return $stmt->fetchAll();

    $stmt->close();
}
```

Permite el llenado de la vista general de un pago obteniendo todos los registros de la base de datos de los pagos para mostrarlo en la interfaz admin del sistema.