Manual Tecnico

Examen Unidad 2

Control de Alumnos

Tecnologias y Aplicaciones Web

Francisco Isaac Perales Morales

controller.php

Este archivo contiene la clase mvcController que tiene los controladores los cuales maneja partes de sistema las cuales no pertenecen a una sección en especifico en el sistema.

El primer controlador que encontraremos es template este controlador incluye en la sección en que estemos la plantilla con todos los estilos y archivos para el funcionamiento del sistema.

```php
<?php

class mvcController
{
    //Control para invocar la platilla con el diseño del sitio
    public function template()
    {
        //incluimos el archivo con la plantilla
        include "views/template.php";
    }
```

El siguiente controlador es urlController este controlador obtiene mediante GET la sección que nos debe desplegar el sistema y la acción que se debe realizar si es que en esa sección estamos realizando algo en especifico y una vez que las obtiene se las enviá urlModel para que nos retorne el archivo con la sección que nos debe desplegar para lo que debamos realizar en ella.

```php
//Control para manejar el redireccionamiento de las distintas secciones del sitio
public function urlController()
{
    //verifica si de debe dirigir a una seccion en especifico con GET
    if(isset($_GET["section"]))
    {
        //se obtiene la seccion de direccionar
        $section = $_GET["section"];
    }
    else
    {
        //en caso de no ser asi se le direccionara al index
        $section = "index";
    }

    //se verifica si en dicha seccion se realizara una accion en especifico
    if(isset($_GET["action"]))
    {
        //se obtiene la accion a realizar
        $action = $_GET["action"];
    }
    else
    {
        //sino entonces no se realizara alguna accion en la seccion
        $action = "index";
    }

    //se llama al modelo utilizado para el direccionaiento
    $url = url::urlModel($section,$action);

    //y se incluye la pagina a la que se va a derireccionar
    include $url;
}
```

El siguiente controlador loginController obtiene el usuario y la contraseña del formulario de inicio de sesión para pasárselo a loginModel el cual nos retornara el usuario registrado en la base de datos que coincida con el usuario ingresado y si su contraseña de los ingresados y devueltos por el modelo se inicia sesión y se lo redirige a la zona de administración del sistema.

```php
//Control para manejar el acceso al sistema
public function loginController()
{
    //se verifica si mediante el formulario se envio informacion
    if(isset($_POST["user"]))
    {
        //se guarda la informacion del login
        $data = array("usuario"=>$_POST["user"],
                      "password"=>$_POST["password"]);

        //se manda al modelo la informacion del login
        $resp = CRUD::loginModel($data,"Usuario");

        //se verifica que la informacion mandada por el formulario sea igual a la devuelta por el modelo
        if($resp["username"] == $_POST["user"]  && $resp["password"] == $_POST["password"])
        {
            //en caso de que coincida se inicia sesion y se guardan cierto datos del usuario
            $_SESSION["nombre"] = $resp["nombre"];
            $_SESSION["password"] = $resp["password"];

            //si no se direccionara al dashboard
            echo "<script>
                    window.location.replace('index.php?section=dashboard');
                  </script>";
        }
    }
}
```

Por ultimo tenemos a infoSistemaController este controlador recibe de infoSistemaModel información de cuantos grupos, alumnos y pagos se tiene registrados en el sistema.

```php
    //contron para obtener la informacion del sistema
    public function infoSistemaController()
    {
        //llamamos al modelo para obtener la informacion de la tienda
        $data = CRUD::infoSistemaModel("Grupo","Alumna","Pago");

        return $data;
    }
}
?>
```

controllerAlumno.php

Este archivo contiene la clase mvcAlumno que tiene los controladores los cuales maneja todo lo relacionado con la sección de Alumnos.

Como primero tenemos el controlador agregarAlumnoController este controlador recibe la información de un alumno del formulario de registro para después mandárselo a agregarAlumnoModel el cual registrara al nuevo alumno, si se registra correctamente nos mandara al listado de alumnos y sino nos mandara un mensaje de error.

```php
<?php

class mvcAlumno
{
    //Control para manejar el registro de un nuevo alumno
    function agregarAlumnoController()
    {
        //se verifica si mediante el formulario se envio informacion
        if(isset($_POST["nombre"]))
        {
            //se guardan la informacion del alumno
            $data = array("nombre" => $_POST["nombre"],
                          "apellido" => $_POST["apellido"],
                          "fecha" => $_POST["fecha"],
                          "grupo" => $_POST["grupo"]);

            //se manda la informacion al modelo con su respectiva tabla en la que se registrara
            $resp = CRUDAlumno::agregarAlumnoModel($data,"Alumna");

            //en caso de que se haya registrado correctamente
            if($resp == "success")
            {
                //asignamos el tipo de mensaje a mostrar
                $_SESSION["mensaje"] = "agregar";

                //nos redireccionara al listado de alumno
                echo "<script>
                        window.location.replace('index.php?section=alumno&action=listado');
                      </script>";
            }
            else
            {
                //sino mandara un mensaje de error
                echo "error";
            }
        }
    }
```

El siguiente controlador es listadoAlumnoController el cual recibe la información de todos los alumnos registrados de listadoAlumnoModel para después mediante un ciclo imprimir su información y los botones de accionen en una fila de un dataTable

```php
//Control para mostrar un listado de los alumnos registrados en el sistema
function listadoAlumnoController()
{
    //se le manda al modelo el nombre de la tabla a mostrar la informacion de los alumnos
    $data = CRUDAlumno::listadoAlumnoModel("Alumna","Grupo");

    //se imprime la informacion de cada uno de los alumnos registrados
    foreach($data as $rows => $row)
    {
        //e imprimimos la informacion de cada uno de los alumnos
        echo "<tr>
            <td>".$row["id_alumna"]."</td>
            <td>".$row["nombre"]." ".$row["apellido"]."</td>
            <td>".$row["fechaNac"]."</td>
            <td>".$row["grupo"]."</td>
            <td>
                <center>
                    <div class='btn-group'>
                            <button type='button' title='Editar Alumno' class='btn btn-default' data-toggle='modal' data-
                            target='#edit-alumno' onclick='idEdit(".$row["id_alumna"].")'>
                                <i class='fa fa-edit'></i>
                            </button>

                        <button type='button' title='Eliminar Alumno' class='btn btn-default' data-toggle='modal' data-
                        target='#del-alumno' onclick='idDel(".$row["id_alumna"].")'>
                            <i class='fa fa-trash-o'></i>
                        </button>
                    </div>
                </center>
            </td>
        </tr>";
    }
}
```

El siguiente es eliminarAlumnoController el cual obtiene del formulario de confirmación de borrado el id del alumno a eliminar para mandárselo a eliminarAlumnoModel junto con las tablas en donde se tiene información del alumno para borrarlo del sistema y se elimino correctamente nos mandara al listado de alumnos y nos mostrara un mensaje.

```php
//Control para borrar un alumno del sistema
public function eliminarAlumnoController()
{
    //se verifica si se envio el id del alumno a eliminar
    if(isset($_POST["del"]))
    {
        //de ser asi se guarda el id del alumno
        $data = $_POST["del"];

        //y se manda al modelo el id y el nombre de la tabla de donde se va a eliminar
        $resp = CRUDAlumno::eliminarAlumnoModel($data,"Pago","Alumna");

        //en caso de haberse eliminado correctamente
        if($resp == "success")
        {
            //asignamos el tipo de mensaje a mostrar
            $_SESSION["mensaje"] = "eliminar";

            //nos redireccionara al listado de alumno
            echo "<script>
                    window.location.replace('index.php?section=alumno&action=listado');
                  </script>";
        }
    }
}
```

Después esta editarAlumnoController el cual recibe el id de un alumno el cual se modificara su información para enviárselo a editarAlumnoController, el cual nos retornara la información del alumnos para después mostrar su información en un formulario para su edición.

```php
//Control para poder mostrar la informacion de un alumno a editar
public function editarAlumnoController()
{
    //se obtiene el id del alumno a mostrar su informacion
    $data = $_POST["edit"];

    //se manda el id del alumno y el nombre de la tabla donde esta almacenada
    $resp = CRUDAlumno::editarAlumnoModel($data,"Alumna");

    //se imprime la informacion del grupo en inputs de un formulario
    echo "
            <input type=hidden value=".$resp["id_alumna"]." name='id'>

            <div class='form-group'>
                    <label>Nombre</label>
                    <input type='text' value='".$resp["nombre"]."' class='form-control' name='nombre'
                    placeholder='Ingrese Nombre' required>
                </div>

                <div class='form-group'>
                    <label>Apellido</label>
                    <input type='text' value='".$resp["apellido"]."' class='form-control' name='apellido'
                    placeholder='Ingrese Apellido' required>
                </div>
```

ModificarAlumnoController recibe la información modificada de un alumno para después mandársela a modificarAlumnoModel el cual actualiza la información del alumno, si la información se actualizo correctamente nos mandara al listado de alumnos.

```php
//Control para modificar la informacion de un alumno
public function modificarAlumnoController()
{
    //se verifica si mediante el formulario se envio informacion
    if(isset($_POST["nombre"]))
    {
        //se guardan la informacion del alumno
        $data = array("id" => $_POST["id"],
                      "nombre" => $_POST["nombre"],
                      "apellido" => $_POST["apellido"],
                      "fecha" => $_POST["fecha"],
                      "grupo" => $_POST["grupo"]);

        //se manda la informacion del alumno y la tabla en la que esta almacenada
        $resp = CRUDAlumno::modificarAlumnoModel($data,"Alumna");

        //en caso de que se haya editado correctamente
        if($resp == "success")
        {
            //asignamos el tipo de mensaje a mostrar
            $_SESSION["mensaje"] = "editar";

            //nos redireccionara al listado de alumnos
            echo "<script>
                    window.location.replace('index.php?section=alumno&action=listado');
                </script>";
        }
    }
}
```

Por ultimo esta infoAlumnoController el cual obtiene la información de los alumnos mediante listadoAlumnoModel para después ingresar la información de los alumnos a un array de JavaScript.

```php
    //Control para obtener las informacion de los alumnos
    public function infoAlumnoController()
    {
        //se le manda al modelo el nombre de la tabla a mostrar su informacion
        $data = CRUDAlumno::listadoAlumnoModel("Alumna","Grupo");

        //mostramos el nombre de cada una de los grupos
        foreach($data as $rows => $row)
        {
            echo "alumnos.push('".$row["id_alumna"].",".$row["nombre"]."
            ".$row["apellido"].",".$row["grupo"]."');";
        }
    }

}
?>
```

controllerPago.php

Este archivo contiene la clase mvcPago que tiene los controladores los cuales maneja todo lo relacionado con la sección de Pagos.

Como primer controlador se encuentra agregarPagoController, el cual obtiene la información de un nuevo pago del formulario de registro para los pagos, también verifica que la imagen enviada por el formulario sea jpg o png, que no pese mas de 5MB y que se pueda mover al servidor, si todo va bien, esta información se la enviá a agregarPagoModel el cual registra el pago en la base de datos, y si se registro correctamente nos redirige al listado de pagos, sino nos manda un mensaje de error.

```php
<?php

class mvcPago
{
    //Control para manejar el registro de un nuevo pago en el sistema
    function agregarPagoController()
    {
        //se verifica si mediante el formulario se envio informacion
        if(isset($_POST["alumno"]))
        {

            //se guardan la informacion del producto
            $data = array("alumno" => $_POST["alumno"],
                          "mama" => $_POST["nomMadre"]." ".$_POST["apeMadre"],
                          "pago" => $_POST["datePago"],
                          "img" => "views/media/img/noimg.png",
                          "folio" => $_POST["folio"]);

            //se verifica si se envio una imagen del boleto
            if(!empty($_FILES["img"]["name"]))
            {
                //se extrae el tipo de la imagen
                $type = $_FILES["img"]["type"];

                //se extrae el tamaño de la imagen
                $size = $_FILES["img"]["size"];
```

listadoPagoAdminController se encarga de mostrar todo los pagos registrados en el sistema, para ello recibe la información de los pagos de listadoPagoModel para después mostrar su información en una fila de un dataTable.

```php
//Control para mostrar un listado de los pagos en la seccion publica
function listadoPagoAdminController()
{
    //se le manda al modelo el nombre de la tabla a mostrar la informacion de los alumnos
    $data = CRUDPago::listadoPagoModel("Pago","Alumna");

    //variable para obtener el lugar en el listado segun la fecha de envio
    $i = 1;

    //se imprime la informacion de cada uno de los alumnos registrados
    foreach($data as $rows => $row)
    {
        //e imprimimos la informacion de cada uno de los alumnos
        echo "<tr>
            <td>".$i."</td>
            <td>".$row["nombre"]." ".$row["apellido"]."</td>
            <td>".$row["mama"]."</td>
            <td>".$row["fecha_pago"]."</td>
            <td>".$row["fecha_envio"]."</td>
            <td><a href='".$row["img_comprobante"]."'>Ver</a></td>
            <td>".$row["folio"]."</td>
            <td>
                <center>
                    <div class='btn-group'>
                            <button type='button' title='Editar Pago' class='btn btn-default' data-
                            toggle='modal' data-target='#edit-pago'
                            onclick='idEdit(".$row["id_pago"].")'>
                                <i class='fa fa-edit'></i>
                            </button>

                        <button type='button' title='Eliminar Pago' class='btn btn-default' data-
                        toggle='modal' data-target='#Del-pago' onclick='idDel(".$row["id_pago"].")'>
                            <i class='fa fa-trash-o'></i>
                        </button>
                    </div>
                </center>
            </td>
        </tr>";

        $i++;
    }
}
```

El siguiente es eliminarPagoController, este controlador recibe el id del pago que se va a eliminar, después le manda a eliminarPagoModel el id del pago a eliminar y si se elimina correctamente nos mandara al listado de pagos.

```php
//Control para borrar un pago del sistema
public function eliminarPagoController()
{
    //se verifica si se envio el id del pago a eliminar
    if(isset($_POST["del"]))
    {
        //de ser asi se guarda el id del pago
        $data = $_POST["del"];

        //y se manda al modelo el id y el nombre de la tabla de donde se va a eliminar
        $resp = CRUDPago::eliminarPagoModel($data,"Pago");

        //en caso de haberse eliminado correctamente
        if($resp == "success")
        {
            //asignamos el tipo de mensaje a mostrar
            $_SESSION["mensaje"] = "eliminar";

            //nos redireccionara al listado de usuarios
            echo "<script>
                    window.location.replace('index.php?section=pago&action=listado');
                  </script>";
        }
    }
}
```

Después editarPagoController, este controlador recibe el id de un pago al cual se le modificara su información, el id se lo manda a editarPagoModel el cual regresa al modelo la información del pago para ser mostrado en un formulario para así ser editada su información.

```php
//Control para poder mostrar la informacion de un pago a editar
public function editarPagoController()
{
    //se obtiene el id del pago a mostrar su informacion
    $data = $_POST["edit"];

    //se manda el id del pago y el nombre de la tabla donde esta almacenada
    $resp = CRUDPago::editarPagoModel($data,"Pago","Alumna");

    //se imprime la informacion del pago en inputs de un formulario
    echo "
            <input type=hidden value=".$resp["id_pago"]." name='id'>

            <div class='form-group'>
                    <label>Nombre</label>
                    <input type='text' value='".$resp["nombre"]." ".$resp["apellido"]."' class='form-
                    control' placeholder='Ingrese Nombre' required readOnly>
                </div>

                <div class='form-group'>
                    <label>Madre</label>
                    <input type='text' value='".$resp["mama"]."' class='form-control' name='mama'
                    placeholder='Nombre Madre' required>
                </div>
```

Por ultimo tenemos a modificarPagoController, el cual recibe la información de un pago la cual fue modificado para ser enviada a modificarPagoModel el cual actualizara la información del pago, si se actualizo correctamente la información nos enviara a el listado de pagos.

```php
    //Control para modificar la informacion de un pago
    public function modificarPagoController()
    {
        //se verifica si mediante el formulario se envio informacion
        if(isset($_POST["id"]))
        {
            //se guardan la informacion del pago
            $data = array("id" => $_POST["id"],
                          "mama" => $_POST["mama"],
                          "pago" => $_POST["pago"],
                          "folio" => $_POST["folio"],
                          "envio" => $_POST["envio"]);

            //se manda la informacion del pa y la tabla en la que esta almacenada
            $resp = CRUDPago::modificarPagoModel($data,"Pago");

            //en caso de que se haya editado correctamente
            if($resp == "success")
            {
                //asignamos el tipo de mensaje a mostrar
                $_SESSION["mensaje"] = "editar";

                //nos redireccionara al listado de pagos
                echo "<script>
                        window.location.replace('index.php?section=pago&action=listado');
                    </script>";
            }
        }
    }
}
?>
```

ControllerGrupo.php

Este archivo contiene la clase mvcGrupo la cual tienen los controladores los cuales manejan todo lo relacionado con la sección de pagos.

El controlador agregarGrupoController recibe la información de un nuevo grupo de su formulario de registro, esta información se la enviá a agregarGrupoModel para registrarlo en la base de datos, si la información del grupo se registra correctamente no mandara a la sección de listado de grupo, sino nos mandara un mensaje de error.

```php
<?php

class mvcGrupo
{
    //Control para manejar el registro de un nuevo grupo
    function agregarGrupoController()
    {
        //se verifica si mediante el formulario se envio informacion
        if(isset($_POST["nombre"]))
        {
            //se guardan la informacion de grupo
            $data = $_POST["nombre"];

            //se manda la informacion al modelo con su respectiva tabla en la que se registrara
            $resp = CRUDGrupo::agregarGrupoModel($data,"Grupo");

            //en caso de que se haya registrado correctamente
            if($resp == "success")
            {
                //asignamos el tipo de mensaje a mostrar
                $_SESSION["mensaje"] = "agregar";

                //nos redireccionara al listado de grupos
                echo "<script>
                        window.location.replace('index.php?section=grupo&action=listado');
                      </script>";
            }
            else
            {
                //sino mandara un mensaje de error
                echo "error";
            }
        }
    }
```

listadoGrupoControles se encarga de mostrar la información de los grupos registrados en el sistema, para ello recibe la información de los grupos de listadoGrupoModel y después imprime la información de los grupos en filas de un dataTable.

```php
//Control para mostrar un listado de los grupos registrados en el sistema
function listadoGrupoController()
{
    //se le manda al modelo el nombre de la tabla a mostrar la informacion de los grupos
    $data = CRUDGrupo::listadoGrupoModel("Grupo");

    //se imprime la informacion de cada uno de los grupos registrados
    foreach($data as $rows => $row)
    {
        //e imprimimos la informacion de cada uno de los grupos
        echo "<tr>
            <td>".$row["id_grupo"]."</td>
            <td>".$row["nombre"]."</td>
            <td>
                <center>
                    <div class='btn-group'>
                            <button type='button' title='Editar Grupo' class='btn btn-default' data-
                            toggle='modal' data-target='#edit-grupo'
                            onclick='idEdit(".$row["id_grupo"].")'>
                                <i class='fa fa-edit'></i>
                            </button>

                        <button type='button' title='Eliminar Grupo' class='btn btn-default' data-
                        toggle='modal' data-target='#del-grupo' onclick='idDel(".$row["id_grupo"].")'>
                            <i class='fa fa-trash-o'></i>
                        </button>
                    </div>
                </center>
            </td>
        </tr>";
    }
}
```

eliminarUsuarioController recibe el id de un grupo el cual se lo manda a eliminarGrupoModel el cual eliminara grupo de la base de datos, si este es eliminado correctamente nos mandara al listado de grupos.

```php
//Control para borrar un Grupo del sistema
public function eliminarGrupoController()
{
    //se verifica si se envio el id del grupo a eliminar
    if(isset($_POST["del"]))
    {
        //de ser asi se guarda el id del grupo
        $data = $_POST["del"];

        //y se manda al modelo el id y el nombre de la tabla de donde se va a eliminar
        $resp = CRUDGrupo::eliminarGrupoModel($data,"Alumna","Grupo");

        //en caso de haberse eliminado correctamente
        if($resp == "success")
        {
            //asignamos el tipo de mensaje a mostrar
            $_SESSION["mensaje"] = "eliminar";

            //nos redireccionara al listado de grupos
            echo "<script>
                    window.location.replace('index.php?section=grupo&action=listado');
                  </script>";
        }
    }
}
```

editarGrupoModel obtiene el id de un grupo al cual se le modificara la información, para esto manda el id a editarGrupoModel el cual le regresa la información para después mostrar la información en un formulario para poder modificarla.

```php
//Control para poder mostrar la informacion de un grupo a editar
public function editarGrupoController()
{
    //se obtiene el id del grupo a mostrar su informacion
    $data = $_POST["edit"];

    //se manda el id del grupo y el nombre de la tabla donde esta almacenada
    $resp = CRUDGrupo::editarGrupoModel($data,"Grupo");

    //se imprime la informacion del grupo en inputs de un formulario
    echo "
            <input type=hidden value=".$resp["id_grupo"]." name='id'>

            <div class='form-group'>
                <label>Nombre</label>
                <input type='text' value='".$resp["nombre"]."' class='form-control' name='nombre'
                placeholder='Ingrese Nombre' required>
            </div>
        ";
}
```

modificarGrupoController recibe la información modificada del grupo y se la manda a modificarGrupoModel el cual actualiza la información del grupo en la base y si se actualiza correctamente nos manda al listado de grupos.

```php
//Control para modificar la informacion de un grupo
public function modificarGrupoController()
{
    //se verifica si mediante el formulario se envio informacion
    if(isset($_POST["nombre"]))
    {
        //se guardan la informacion del grupo
        $data = array("id" => $_POST["id"], "nombre" => $_POST["nombre"]);

        //se manda la informacion del grupo y la tabla en la que esta almacenada
        $resp = CRUDGrupo::modificarGrupoModel($data,"Grupo");

        //en caso de que se haya editado correctamente
        if($resp == "success")
        {
            //asignamos el tipo de mensaje a mostrar
            $_SESSION["mensaje"] = "editar";

            //nos redireccionara al listado de grupos
            echo "<script>
                    window.location.replace('index.php?section=grupo&action=listado');
                  </script>";
        }
    }
}
```

Por ultimo esta optionGrupoController el cual recibe la información de los grupos mediante listadoGrupoModel para después desplegar su información mediante options de un select2.

```php
    //Control para mostrar los grupos en un select
    public function optionGrupoController()
    {
        //se le manda al modelo el nombre de la tabla a mostrar su informacion
        $data = CRUDGrupo::listadoGrupoModel("Grupo");

        //mostramos el nombre de cada una de los grupos
        foreach($data as $rows => $row)
        {
            //se muestra cada una de los grupos en un option del select
            echo "<option value=".$row["id_grupo"].">".$row["nombre"]."</option>";
        }
    }
}
?>
```

conexion.php

Este archivo unicamente contiene un la clase conexión con el cual se crea conexiones a la base de datos para poder realizar alguna operación en ella.

El único modelo que tiene es conectar, este modelo crea una conexión a la base de daos mediante PDO y la retorna al modelo la cual la solicite.

```php
<?php

//clase para realizar la conexion a la base de datos
class conexion
{
    //funcion para retornar una conexion a la base de datos
    public static function conectar()
    {
        //creamos la conexion
        $conn = new PDO("mysql:host=localhost;dbname=curso","root","");

        //y la retornamos
        return $conn;
    }
}
?>
```

url.php

Este archivo contiene la clase de url la cual gestiona el direccionamiento del sitio.

Esta clase contiene el modelo urlModel el cual utiliza la la sección y la acción recibida para el controlador para después retornar al controlador la url del archivo de la sección que se debe incluir para mostrarse en el navegador.

```php
<?php
//clase para realizar el redireccionamiento del sitio
class url
{
    //modelo para realizar el redireccionamiento del sitio
    public static function urlModel($section,$action)
    {
        //en caso de que se mande un link valido se redirecciona a su pagina correspondiene
        if(($section == "grupo" || $section == "alumno" ||  $section == "pago") && ($action ==
        "agregar" || $action == "listado" || $action == "eliminar" || $action == "editar"))
        {
            $url = "views/".$section."/".$action.".php";
        }
        elseif($section == "listado" || $section == "dashboard"  || $section == "logout")
        {
            $url = "views/modules/".$section.".php";
        }
        elseif($section == "admin")
        {
            $url = "views/modules/login.php";
        }
        else
        {
            $url = "views/modules/registro.php";
        }

        //y se retorna la pagina a redireccionar
        return $url;
    }
}
?>
```

crud.php

Este archivo contiene la clase CRUD la cual contiene los modelos para el control de las operaciones de la base de datos que no pertenecen a una sección en especifica del sitio.

El primer modelo que contiene es loginModel, este modelo recibe la tabla y los datos de un formulario de inicio de sesión, con ellos realiza una consulta a la base de datos para averiguar si existe en la base de datos un registro con el nombre de usuario ingresado. Después de realizar la consulta retorna el resultado al modelo y cierra la conexión a la base de datos

```php
<?php
require_once "conexion.php";

//clase para realizar operaciones a la base de datos
class CRUD
{
    //modelo para obtener la informacion para el login
    public static function loginModel($data,$tabla)
    {
        //preparamos la sentencia para realizar el select
        $stmt = Conexion::conectar()->prepare("SELECT * FROM $tabla WHERE username = :id");

        //se realiza la asignacion de los datos para la consulta
        $stmt->bindParam(":id", $data["usuario"], PDO::PARAM_STR);

        //se ejecuta la sentencia
        $stmt->execute();

        //retornamos la fila obtenida con el select
        return $stmt->fetch();

        //cerramos la conexion
        $stmt->close();
    }
```

Por ultimo esta infoSistemaModel, este modelo realiza una serie de consultas a la base de datos para obtener información de cuantos registro tiene de las tablas las cuales el controlador les mando como parámetro, después de realizar una consulta guarda la información de la consulta en un array, el cual una vez realizadas todas las consultas es retornado al controlador

```php
//modelo para obtener la informacion del sistema
public static function infoSistemaModel($tabla1,$tabla2,$tabla3)
{
    $info = array();

    //preparamos la consulta para obtener la cantidad de grupos registrados
    $stmt = Conexion::conectar() -> prepare("SELECT COUNT(*) as grupos FROM $tabla1");

    //ejecutamos la consulta
    $stmt -> execute();

    //obtenemos el valor devuelto por la consulta
    $res = $stmt -> fetch();

    //y la almacenamos en info
    $info["grupo"] = $res["grupos"];


    //preparamos la consulta para obtener a los alumnos registrados
    $stmt = Conexion::conectar() -> prepare("SELECT COUNT(*) as alumnos FROM $tabla2");
```

crudAlumno.php

Este archivo contiene la clase CRUDAlumno la cual contiene los modelos utilizados para las operaciones de base de datos para la sección de Alumno.

El primer modelo que contiene es agregarAlumnoModel este modelo recibe del controlador la información de un nuevo alumno para ser registrado en la base de datos y la tabla en donde sera registrado, para ello crea una conexión a la base de datos para preparar un insert, después asigna la información del alumno y ejecuta el insert si se realizo exitosamente el insert retornara al controlador success, en caso contrario retornara fail.

```php
<?php
require_once "conexion.php";

//clase para realizar operaciones a la base de datos para la seccion de alumnos
class CRUDAlumno
{
    //modelo para registrar un alumno en la base de datos
    public static function agregarAlumnoModel($data,$tabla)
    {
        //se prepara la sentencia para realizar el insert
        $stmt = Conexion::conectar() -> prepare("INSERT INTO $tabla (nombre,apellido,fechaNac,grupo)
        VALUES (:nombre,:apellido,:fecha,:grupo)");


        //cambiamos el formato de la fecha a yyyy/mm/dd
        $fecha = date("Y/m/d", strtotime($data["fecha"]));

        //se realiza la asignacion de los datos a insertar
        $stmt -> bindParam(":nombre",$data["nombre"],PDO::PARAM_STR);
        $stmt -> bindParam(":apellido",$data["apellido"],PDO::PARAM_STR);
        $stmt -> bindParam(":fecha",$fecha,PDO::PARAM_STR);
        $stmt -> bindParam(":grupo",$data["grupo"],PDO::PARAM_STR);

        //se ejecuta la sentencia
        if($stmt -> execute())
        {
            //si se ejecuto correctamente nos retorna success
            return "success";
        }
        else
        {
            //en caso de no ser asi nos retorna fail
            return "fail";
        }

        print_r($data);

        //cerramos la conexion
        $stmt -> close();
    }
```

El modelo listadoAlumnoModel realiza una conexión a la base de datos para preparar una consulta a las tablas recibidas por el controlador para obtener la información de todos los alumnos registrados en el sistema, después se ejecuta la consulta y retorna al controlador toda la información obtenida de la consulta y por ultimo cierra la conexión.

```php
//modelo para obtener la informacion de los alumnos registrados
public static function listadoAlumnoModel($tabla1,$tabla2)
{
    //preparamos la consulta
    $stmt = Conexion::conectar() -> prepare("SELECT a.id_alumna, a.nombre, a.apellido,
    a.fechaNac, g.nombre as grupo FROM $tabla1 as a JOIN $tabla2 as g on a.grupo = g.id_grupo");

    //se ejecuta la consulta
    $stmt -> execute();

    //retornamos la informacion de la tabla
    return $stmt -> fetchAll();

    //cerramos la conexion
    $stmt -> close();
}
```

El modelo eliminarAlumnoModel se utiliza para eliminar la información de un alumno de la base de datos, para ello recibe del controlador el id del alumnos y las tablas de donde se debe borrar su información, para ello realiza 2 conexiones a la base de datos para prepara un delete en donde se le asigna el id del alumno para solamente borrar la información de ese alumno. Después se ejecutan las consultas, si se ejecuta correctamente el modelo nos retornara success y sino fail, por ultimo cierra la conexión.

```php
//modelo para borrar un alumno de la base de datos
public static function eliminarAlumnoModel($data,$tabla1,$tabla2)
{
    //preparamos la sentencia para realizar el Delete
    $stmt1 = Conexion::conectar() -> prepare("DELETE FROM $tabla1 WHERE alumna = :id");

    //se realiza la asignacion de los datos a eliminar
    $stmt1 -> bindParam(":id",$data,PDO::PARAM_INT);

    //preparamos la sentencia para realizar el delete
    $stmt2 = Conexion::conectar() -> prepare("DELETE FROM $tabla2 WHERE id_alumna = :id");

    //se realiza la asignacion de los datos a eliminar
    $stmt2 -> bindParam(":id",$data,PDO::PARAM_INT);

    //se ejecuta las sentencias
    if($stmt1 -> execute() && $stmt2 -> execute())
    {
        //si se ejecuto correctamente nos retorna success
        return "success";
    }
    else
    {
        //en caso de no ser asi nos retorna fail
        return "fail";
    }

    //cerramos la conexion
    $stmt -> close();
}
```

editarAlumnoModel se utiliza para obtener la información de un alumno, para ello recibe del controlador y tabla de donde se extraerá su información, para ello crea una conexión a la base de datos para preparar un select después al select se le asigna el id del alumno al cual se desea obtener su información, después se ejecuta la consulta y se retorna la información obtenida y por ultimo se cierra la conexión.

```php
//modelo para obtener la informacion de un alumno
public static function editarAlumnoModel($data,$tabla)
{
    //preparamos la sentencia para realizar el select
    $stmt = Conexion::conectar()->prepare("SELECT * FROM $tabla WHERE id_alumna = :id");

    //se realiza la asignacion de los datos para la consulta
    $stmt->bindParam(":id",$data, PDO::PARAM_INT);

    //se ejecuta la sentencia
    $stmt->execute();

    //retornamos la fila obtenida con el select
    return $stmt->fetch();

    //cerramos la conexion
    $stmt->close();
}
```

Por ultimo esta modificarAlumnoModel el cual se utiliza para la modificación de la información de un alumno, para esto recibe la información actualizada y la tabla en la que se actualizara la información del controlador, después crea una conexión a la base de datos para crear un update, después asignara la información actualizada a las variables del update y ejecutara el update, si se ejecuto correctamente nos retornara un success sino nos retornada un fail y por ultimo cierra la conexión.

```php
    //modelo para modificar la informacion de un alumno registrada en la base de datos
    public static function modificarAlumnoModel($data,$tabla)
    {
        //preparamos la sentencia para realizar el update
        $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre, apellido =
        :apellido, fechaNac = :fecha, grupo = :grupo WHERE id_alumna = :id");

        //cambiamos el formato de la fecha a yyyy/mm/dd
        $fecha = date("Y/m/d", strtotime($data["fecha"]));

        //se realiza la asignacion de los datos para el update
        $stmt -> bindParam(":id", $data["id"], PDO::PARAM_INT);
        $stmt -> bindParam(":nombre", $data["nombre"], PDO::PARAM_STR);
        $stmt -> bindParam(":apellido", $data["apellido"], PDO::PARAM_STR);
        $stmt -> bindParam(":fecha", $fecha, PDO::PARAM_STR);
        $stmt -> bindParam(":grupo", $data["grupo"], PDO::PARAM_STR);

        //se ejecuta la sentencia
        if($stmt -> execute())
        {
            //si se ejecuto correctamente nos retorna success
            return "success";
        }
        else
        {
            print_r($stmt -> errorInfo());
            //en caso de no ser asi nos retorna fail
            return "fail";
        }

        //cerramos la conexion
        $stmt->close();
    }
}
```

crudPago.php

Este archivo contiene la clase CRUDPago la cual contiene los modelos para manejar las operaciones de la base de datos para la sección de Pagos.

Como primer modelo tenemos agregarPagoModel, este modelo recibe del controlador la información de un nuevo para a ser registrado, para ello realiza una conexión a la base de datos para prepara un insert, después asigna los datos del pago a las variables del insert y ejecuta la consulta, si esta se ejecuto correctamente retornara al modelo success, en caso contrario nos retornara un fail y por ultimo cerrara la conexión.

```php
<?php
require_once "conexion.php";

//clase para realizar operaciones a la base de datos para la seccion de alumnos
class CRUDPago
{
    //modelo para registrar un alumno en la base de datos
    public static function agregarPagoModel($data,$tabla)
    {
        //se prepara la sentencia para realizar el insert
        $stmt = Conexion::conectar() -> prepare("INSERT INTO $tabla
        (alumna,mama,fecha_pago,fecha_envio,img_comprobante,folio) VALUES
        (:alumna,:mama,:fecha_pago,NOW(),:img,:folio)");


        //cambiamos el formato de la fecha a yyyy/mm/dd
        $fecha = date("Y/m/d", strtotime($data["pago"]));

        //se realiza la asignacion de los datos a insertar
        $stmt -> bindParam(":alumna",$data["alumno"],PDO::PARAM_INT);
        $stmt -> bindParam(":mama",$data["mama"],PDO::PARAM_STR);
        $stmt -> bindParam(":fecha_pago",$fecha,PDO::PARAM_STR);
        $stmt -> bindParam(":img",$data["img"],PDO::PARAM_STR);
        $stmt -> bindParam(":folio",$data["folio"],PDO::PARAM_INT);

        //se ejecuta la sentencia
        if($stmt -> execute())
        {
            //si se ejecuto correctamente nos retorna success
            return "success";
        }
        else
        {
            //en caso de no ser asi nos retorna fail
            return "fail";
        }

        //cerramos la conexion
        $stmt -> close();
    }
```

El modelo listadoPagoModel se utiliza para obtener la información de todos los pago registrados en el sistema, para ello recibe del controlador las tablas de donde se extraerá la información, crea una conexión para preparar un consulta a las tablas, después ejecuta la consulta y retorna la información obtenida y por ultimo cierra la conexión.

```php
//modelo para obtener la informacion de los Pagos registrados
public static function listadoPagoModel($tabla1,$tabla2)
{
    //preparamos la consulta
    $stmt = Conexion::conectar() -> prepare("SELECT
    p.id_pago,a.nombre,a.apellido,p.fecha_envio,p.fecha_pago,p.img_comprobante,p.folio,p.mama
    FROM $tabla1 as p JOIN $tabla2 as a on p.alumna = a.id_alumna ORDER BY p.fecha_envio ASC");

    //se ejecuta la consulta
    $stmt -> execute();

    //retornamos la informacion de la tabla
    return $stmt -> fetchAll();

    //cerramos la conexion
    $stmt -> close();
}
```

El modelo eliminarPagoModel se utiliza para eliminar un pago registrado en el sistema, para esto recibe del controlador el id del pago a eliminar y la tabla de donde se elimina, primero realiza una conexión a la base de datos para preparar un delete al cual se le asigna el id del pago a eliminar, después de esto se ejecutara el delete y si se ejecuto correctamente nos devolverá un success y sino nos retornar un fail, por ultimo cierra la conexión.

```php
//modelo para borrar un pago de la base de datos
public static function eliminarPagoModel($data,$tabla)
{
    //preparamos la sentencia para realizar el delete
    $stmt = Conexion::conectar() -> prepare("DELETE FROM $tabla WHERE id_pago = :id");

    //se realiza la asignacion de los datos a eliminar
    $stmt -> bindParam(":id",$data,PDO::PARAM_INT);

    //se ejecuta la sentencia
    if($stmt -> execute())
    {
        //si se ejecuto correctamente nos retorna success
        return "success";
    }
    else
    {
        //en caso de no ser asi nos retorna fail
        return "fail";
    }

    //cerramos la conexion
    $stmt -> close();
}
```

El modelo editarPagoModel se usa para obtener la información de un pago al cual se modificara su información, para esto recibe del controlador el id del pago y las tablas de donde se extraerá la información, primero realiza una conexión a la base de datos para crear una consulta, después se asigna el id del pago a la consulta y de ejecuta, por ultimo retorna la información obtenida y cierra a conexión.

```php
//modelo para obtener la informacion de un pago
public static function editarPagoModel($data,$tabla1,$tabla2)
{
    //preparamos la sentencia para realizar el select
    $stmt = Conexion::conectar()->prepare("SELECT p.id_pago, p.fecha_envio, p.folio,
    p.fecha_pago, p.mama, a.nombre, a.apellido FROM $tabla1 as p JOIN $tabla2 as a on
    a.id_alumna = p.alumna WHERE p.id_pago = :id");

    //se realiza la asignacion de los datos para la consulta
    $stmt->bindParam(":id",$data, PDO::PARAM_INT);

    //se ejecuta la sentencia
    $stmt->execute();

    //retornamos la fila obtenida con el select
    return $stmt->fetch();

    //cerramos la conexion
    $stmt->close();
}
```

Por ultimo modificarPagoModel se usa para actualizar la información de un pago, recibe del controlador la información actualizada del pago y ta tabla en donde se va a actualizar, para esto crea una conexión a la base de datos para prepara un update, después se le asigna al update la información actualizada del pago y se ejecuta, después si se ejecuto correctamente nos retornara un success, en caso contrario un fail y por ultimo cierra la conexión;

```php
    //modelo para modificar la informacion de un pago registrado en la base de datos
    public static function modificarPagoModel($data,$tabla)
    {
        //preparamos la sentencia para realizar el update
        $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET fecha_envio = :envio, mama = :mama,
        fecha_pago = :fecha, folio = :folio WHERE id_pago = :id");

        //cambiamos el formato de la fecha a yyyy/mm/dd
        $fecha = date("Y/m/d", strtotime($data["pago"]));

        //se realiza la asignacion de los datos para el update
        $stmt -> bindParam(":id", $data["id"], PDO::PARAM_INT);
        $stmt -> bindParam(":mama", $data["mama"], PDO::PARAM_STR);
        $stmt -> bindParam(":fecha", $fecha, PDO::PARAM_STR);
        $stmt -> bindParam(":folio", $data["folio"], PDO::PARAM_INT);
        $stmt -> bindParam(":envio", $data["envio"], PDO::PARAM_STR);

        //se ejecuta la sentencia
        if($stmt -> execute())
        {
            //si se ejecuto correctamente nos retorna success
            return "success";
        }
        else
        {
            //en caso de no ser asi nos retorna fail
            return "fail";
        }

        //cerramos la conexion
        $stmt->close();
    }
}
?>
```

crudGrupo.php

Este archivo contiene la clase CRUDGrupo esta clase contiene todos los modelos para manejar las operaciones en la base de datos para la sección de Grupos.

Primero tenemos al modelo agregarGrupoModel, este modelo recibe del controlador los datos de un nuevo grupo a registrase, para esto crea una conexión a la base de datos para prepara un insert, después se le agregara al insert los datos del grupo a registrar para después ejecutar el insert, si esta se ejecuta correctamente nos retornara un success y en caso contrario un fail. Por ultimo cierra la conexión.

```php
<?php
require_once "conexion.php";

//clase para realizar operaciones a la base de datos para la seccion de grupos
class CRUDGrupo
{
    //modelo para registrar un grupo en la base de datos
    public static function agregarGrupoModel($data,$tabla)
    {
        //se prepara la sentencia para realizar el insert
        $stmt = Conexion::conectar() -> prepare("INSERT INTO $tabla (nombre) VALUES (:nombre)");

        //se realiza la asignacion de los datos a insertar
        $stmt -> bindParam(":nombre",$data,PDO::PARAM_STR);

        //se ejecuta la sentencia
        if($stmt -> execute())
        {
            //si se ejecuto correctamente nos retorna success
            return "success";
        }
        else
        {
            //en caso de no ser asi nos retorna fail
            return "fail";
        }

        //cerramos la conexion
        $stmt -> close();
    }
```

El modelo listadoGrupoModel obtiene la información de todos los grupos registrados en el sistema, para esto crea una conexión a la base de datos para preparar una consulta para después ejecutarla y retornar toda la información de los grupos al controlador, por ultimo cierra la conexión.

```php
//modelo para obtener la informacion de los grupos registrados
public static function listadoGrupoModel($tabla)
{
    //preparamos la consulta
    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla");

    //se ejecuta la consulta
    $stmt -> execute();

    //retornamos la informacion de la tabla
    return $stmt -> fetchAll();

    //cerramos la conexion
    $stmt -> close();
}
```

El modelo elimianarGrupoModel elimina un grupo de la base de datos, para realizar esto del controlador recibe el id del grupo y las tablas de donde se va a eliminar, primero crea 2 conexiones a la base de datos una para realizar un update y otra para realizar un delete y se le asigna a ambos el id del grupo a eliminar y se ejecutan, si se ejecutaron correctamente nos retornara success y sino nos retornara fail, por ultimo cierra la conexión.

```php
//modelo para borrar un grupo de la base de datos
public static function eliminarGrupoModel($data,$tabla1,$tabla2)
{
    //preparamos la sentencia para quitar del grupo a las alumnas que esten en el
    $stmt1 = Conexion::conectar() -> prepare("UPDATE $tabla1 SET grupo = NULL WHERE grupo =
    :id");

    //se realiza la asignacion de los datos a actualizar
    $stmt1 -> bindParam(":id",$data,PDO::PARAM_INT);

    //preparamos la sentencia para realizar el delete
    $stmt2 = Conexion::conectar() -> prepare("DELETE FROM $tabla2 WHERE id_grupo = :id");

    //se realiza la asignacion de los datos a eliminar
    $stmt2 -> bindParam(":id",$data,PDO::PARAM_INT);

    //se ejecuta la sentencia
    if($stmt1 -> execute() && $stmt2 -> execute())
    {
        //si se ejecuto correctamente nos retorna success
        return "success";
    }
    else
    {
        //en caso de no ser asi nos retorna fail
        return "fail";
    }

    //cerramos la conexion
    $stmt -> close();
}
```

El modelo editarGrupoModel se usa para obtener la información de un grupo editar su información, para esto recibe del controlador el id del grupo y la tabla en donde esta la información, después crea una conexión a la base de datos para realizar una consulta y se le asigna a la consulta el id del grupo. Después se ejecuta la consulta y se retorna la información, por ultimo cerramos la conexión.

```php
//modelo para obtener la informacion de un grupo
public static function editarGrupoModel($data,$tabla)
{
    //preparamos la sentencia para realizar el select
    $stmt = Conexion::conectar()->prepare("SELECT * FROM $tabla WHERE id_grupo = :id");

    //se realiza la asignacion de los datos para la consulta
    $stmt->bindParam(":id",$data, PDO::PARAM_INT);

    //se ejecuta la sentencia
    $stmt->execute();

    //retornamos la fila obtenida con el select
    return $stmt->fetch();

    //cerramos la conexion
    $stmt->close();
}
```

Por ultimo tenemos a modificarGrupoModel este modelo recibe la información modificada de un grupo y la tabla en donde va a ser modificada, para esto realiza una conexión a la base de datos para preparar un update y después se le asigna la información modificada del grupo, después se ejecuta el update y si se ejecuto correctamente nos retornara un success y si no nos retornara un fail, por ultimo cierra la conexión.

```php
    //modelo para modificar la informacion de un grupo registrada en la base de datos
    public static function modificarGrupoModel($data,$tabla)
    {
        //preparamos la sentencia para realizar el update
        $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre WHERE id_grupo =
        :id");

        //se realiza la asignacion de los datos para el update
        $stmt -> bindParam(":id", $data["id"], PDO::PARAM_INT);
        $stmt -> bindParam(":nombre", $data["nombre"], PDO::PARAM_STR);

        //se ejecuta la sentencia
        if($stmt -> execute())
        {
            //si se ejecuto correctamente nos retorna success
            return "success";
        }
        else
        {
            //en caso de no ser asi nos retorna fail
            return "fail";
        }

        //cerramos la conexion
        $stmt->close();
    }
}
?>
```

template.php

Este archivo contiene todo el diseño del sitio, en el se incluyen todos los todos los estilos y archivos para que las secciones del sistema funcionen y se visualicen correctamente en el navegador.

```php
<?php
error_reporting(E_ALL);
ini_set("display_errors", 1);

session_start();
?>
<html>
    <head>
        <!--importacion de los archivos con los estilos del template-->
        <meta charset="utf-8">
        <link rel="icon" href="views/media/img/favicon.png" sizes="16x16">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Control de Alumnos</title>
        <!-- Tell the browser to be responsive to screen width -->
        <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no"
        name="viewport">
        <!-- Bootstrap 3.3.7 -->
        <link rel="stylesheet" href="views/media/bower_components/bootstrap/dist/css/bootstrap.min.css">
        <!-- Font Awesome -->
        <link rel="stylesheet" href="views/media/bower_components/font-awesome/css/font-awesome.min.css">
        <!-- Ionicons -->
        <link rel="stylesheet" href="views/media/bower_components/Ionicons/css/ionicons.min.css">
        <!-- Theme style -->
        <link rel="stylesheet" href="views/media/dist/css/AdminLTE.min.css">
        <!-- AdminLTE Skins. Choose a skin from the css/skins folder instead of downloading all of them to
        reduce the load. -->
        <link rel="stylesheet" href="views/media/dist/css/skins/_all-skins.min.css">
        <!-- Morris chart -->
        <link rel="stylesheet" href="views/media/bower_components/morris.js/morris.css">
        <!-- jvectormap -->
        <link rel="stylesheet" href="views/media/bower_components/jvectormap/jquery-jvectormap.css">
        <!-- Date Picker -->
        <link rel="stylesheet" href="views/media/bower_components/bootstrap-datepicker/dist/css/bootstrap-
        datepicker.min.css">
```

login.php

Este archivo contiene el formulario para el inicio de sesión para acceder a la sección de administración del sistema. Ademas se crea un objeto de mvcController para manda a llamar al controlador loginController para cuando se envié información de acceso al sistema mediante el formulario.

```php
<?php

//creamos un objeto de mvcController
$login = new mvcController();

//se manda a llamar el control para manejar el inicio de sesion
$login -> loginController();
?>

<body class="hold-transition login-page">
    <!--caja para mostrar el formulario de inicio de sesion-->
    <div class="login-box">
        <div class="login-logo">
            <h1>Control de Alumnos</h1>
        </div>
        <!-- /.login-logo -->
        <div class="login-box-body">
            <p class="login-box-msg">
                Iniciar Sesion
            </p>

            <!--formulario para ingresar la informacion para que un usuario pueda iniciar sesion en el
            sistema-->
            <form method="post" autocomplete="off">

                <div class="form-group has-feedback">
                    <input type="text" class="form-control" placeholder="Ingrese Usuario" name="user"
                    required>
                    <span class="glyphicon glyphicon-envelope form-control-feedback"></span>
                </div>
                <div class="form-group has-feedback">
                    <input type="password" class="form-control" placeholder="Ingrese Contraseña"
                    name="password" required>
                    <span class="glyphicon glyphicon-lock form-control-feedback"></span>
                </div>
                <div class="row">
                    <!-- /.col -->
                    <div class="col-xs-12">
                        <button type="submit" class="btn btn-primary btn-block btn-flat">Iniciar
                        Sesion</button>
                    </div>
                    <!-- /.col -->
                </div>
            </form>
        </div>
```

logout.php

Este archivo contiene un unicamente un pequeño código el cual destruye la sesión para cerrar la sesión del usuario actual y sacarlo a la sección publica del sistema

```php
<?php
//destruimos la sesion para cerrar la sesion del usuario
session_destroy();

//despues lo redireccionamos al index
echo"<script>
        window.location.replace('index.php');
    </script>";
?>
```

menu.php

Este archivo contiene los menús que se desplegara a los usuarios del sistema, se desplegara uno u otro dependiendo si el usuario esta logeado en el sistema o no.

```php
<!-- Collect the nav links, forms, and other content for toggling -->
<div class="collapse navbar-collapse pull-left" id="navbar-collapse">
    <ul class="nav navbar-nav">
        <?php
        //verificamos si esta logeado para mostrar las secciones a las que tiene acceso
        if(isset($_SESSION["nombre"]))
        {
        ?>
            <!--secciones a las que el usuario tiene acceso si esta logeado-->
            <li><a href="index.php?section=dashboard">Dashboard<span class="sr-only">(current)</span></a>
            </li>
            <li><a href="index.php?section=alumno&action=listado">Alumnos</a></li>
            <li><a href="index.php?section=grupo&action=listado">Grupos</a></li>
            <li><a href="index.php?section=pago&action=listado">Pagos</a></li>
        <?php
        }
        else
        {
        ?>
            <!--secciones a las que el usuario tiene acceso si no esta logeado-->
            <li><a href="index.php?section=registro">Registro<span class="sr-only">(current)</span></a>
            </li>
            <li><a href="index.php?section=listado">Listados</a></li>
        <?php
        }
        ?>
    </ul>
</div>
```

dashboard.php

Este archivo contiene el código para desplegar información acerca de los registro del sistema, para obtener esta información crea un objeto de mvcController y manda a llamar el controlador infoSistemaController para obtener la información, para después imprimirlo en su widget correspondiente.

```
<section class="content-header">
    <h1>
        Dashboard
    </h1>
    <ol class="breadcrumb">
        <li><a href="#"><i class="fa fa-dashboard"></i>Dashboard</a></li>
    </ol>
</section>

<!-- Main content -->
<section class="content">

    <?php
    //creamos un objeto de mvcController
    $info = new mvcController();

    //llamos al control para mostrar la informacion del sistema
    $data = $info -> infoSistemaController();
    ?>

    <div class="row">
        <div class="col-md-4 col-sm-6 col-xs-12">
            <div class="info-box">
                <span class="info-box-icon bg-aqua"><i class="fa fa-envelope-o"></i></span>

                <div class="info-box-content">
                    <span class="info-box-text">Grupos</span>
                    <span class="info-box-number"><?php echo $data["grupo"]; ?></span>
                </div>
                <!-- /.info-box-content -->
            </div>
            <!-- /.info-box -->
        </div>
        <!-- /.col -->
        <div class="col-md-4 col-sm-6 col-xs-12">
            <div class="info-box">
                <span class="info-box-icon bg-green"><i class="fa fa-flag-o"></i></span>

                <div class="info-box-content">
                    <span class="info-box-text">Alumnos</span>
                    <span class="info-box-number"><?php echo $data["alumno"]; ?></span>
                </div>
                <!-- /.info-box-content -->
            </div>
```

registro.php

En esta archivo contiene el formulario de registro para un pago, en el cual se nos pedira la información para poder registrar un nuevo pago, una vez ingresada la información, se nos enviá a la parte de agregar pago para registrar en nuevo pago.

Y por ultimo nos manda al listado de pagos de la parte publica.

```
<!-- Full Width Column -->
<div class="container">
    <!-- Content Header (Page header) -->
    <section class="content-header">
        <h1>
            Festival 2018
        </h1>
    </section>

    <!-- Main content -->
    <section class="content">
        <div class="row">
            <div class="col-md-12">

                <?php
                //verificamos si se va a mostrar un mensaje de aviso al suceder un error
                if(!empty($_SESSION["error"]))
                { ... }
                ?>

                <div class="box box-solid">
                    <div class="box-header with-border">
                        <h3 class="box-title">Formulario de Envio de Comprobantes</h3>
                    </div>
                    <!-- /.box-header -->
                    <div class="box-body">
                        <form role="form" enctype="multipart/form-data" method="post"
                        action="index.php?section=pago&action=agregar">
                            <div class="box-body">

                                <div class="form-group">
                                    <label>Grupo</label>
                                    <select id="grupo" name="grupo" class="form-control select2"
                                    style="width: 100%;" onchange="Alumno()">
                                        <option value=0 selected="selected">Seleccione
                                        Grupo</option>
                                        <?php
                                        //creamos un objeto de mvcGrupo
                                        $grupo = new mvcGrupo();

                                        //llamamos al controller para traer los grupos en options de
                                        un select
                                        $grupo -> optionGrupoController();
                                        ?>
                                    </select>
```

listado.php

En este archivo se muestra todo los registro ya sea de pagos, alumnos o grupos dependiendo de la sección en que estemos en el sistema, para ello primero verifica si hemos iniciado sesión en el sistema.

Si estamos logeados creada un objeto del la clase correspondiente a la sección en la que estemos y con ese objeto llamara al controlador de listado de la clase para traer la información correspondiente a la sección y mostrar los datos obtenidos en un dataTable.

```php
<?php
//verificamos si el usuario ya ha iniciado session
if(!isset($_SESSION["nombre"]))
{
    //si no ha iniciado sesion, lo sacamos a la seccion publica
    echo "<script>
            window.location.replace('index.php');
          </script>";
}
?>
<!-- Full Width Column -->
<div class="container">
    <!--section para mostrar al Usuario el lugar donde se encuentra-->
    <section class="content-header">
        <h1>
            Grupos
        </h1>
        <ol class="breadcrumb">
            <li><a href="#"><i class="fa fa-dashboard"></i>Inicio</a></li>
            <li class="active">Grupos</li>
        </ol>
    </section>

    <!-- Main content -->
    <section class="content">
        <div class="row">
            <div class="col-md-12">

                <?php
                //verificamos si se va a mostrar un mensaje de aviso al realizar alguna operacion de
                crud
                if(!empty($_SESSION["mensaje"]))
                { ··· }

                ?>

                <!-- caja para mostrar el listado de grupos-->
                <div class="box box-success">
                    <div class="box-header">
                        <div class="row">
                            <div class="col-md-6">
                                <h3 class="box-title">Listado de Grupos</h3>
                            </div>
                            <div class="col-md-6">
                                <button type="button" class="btn btn-success pull-right" data-
```

agregar.php

Este archivo nos despliega un modal con un formulario de registro, para agregar así un nuevo registro a la sección en la que nos encontramos.

Una vez que nos hayamos ingresamos la información nos mandara a la parte de agregar y el archivo creara un objeto de la clase correspondiente a la sección, para mandar a llamar al controlador de agregar, para así registrar la información a la base de datos

```php
<?php
//verificamos si el usuario ya ha iniciado session
if(!isset($_SESSION["nombre"]))
{
    //si no ha iniciado sesion, lo sacamos a la seccion publica
    echo "<script>
            window.location.replace('index.php');
          </script>";
}

//verificamos si se debe llamar al controller para agregar un nuevo grupo
if(isset($_GET["action"]) && $_GET["action"]=="agregar")
{
    //se crea un objeto de mvcGrupo
    $agregar = new mvcGrupo();

    //se manda a llamar el controller para agregar un nuevo Grupo al sistema
    $agregar -> agregarGrupoController();
}
?>

<!--modal para agregar un nuevo Grupo-->
<div class="modal modal-info fade" id="new_grupo">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
                <h4 class="modal-title">Nuevo Grupo</h4>
            </div>

            <!--Formulario para introducir los datos del nuevo grupo-->
            <form role="form" method="post" autocomplete="off" action="index.php?
            section=grupo&action=agregar">
                <div class="modal-body">

                    <div class="form-group">
                        <label>Nombres</label>
                        <input type="text" class="form-control" name="nombre" placeholder="Ingrese
                        Nombre" required>
                    </div>
                </div>
```

editar.php

Este archivo primero nos muestra un modal con un formulario de confirmación, en la cual nos pide que ingresemos la contraseña del usuario actual, después de esto nos muestra un modal con un formulario en donde se muestra dependiendo la sección la información del registro para poder modificarla.

Una vez que ya se modifique la información del registro nos enviá a la parte de agregar y el archivo creara un objeto de la clase correspondiente para mandar a llamar al controlador de modificar de la clase para modificar la información del registro en el sistema.

```php
<?php
//verificamos si el usuario ya ha iniciado session
if(!isset($_SESSION["nombre"]))
{
    //si no ha iniciado sesion, lo redireccionara a la seccion publica
    echo "<script>
            window.location.replace('index.php');
          </script>";
}

//verificamos si se debe mandar a llamar el controller para modificar un grupo
if(isset($_GET["action"]) && $_GET["action"]=="editar")
{
    //creamos un objeto de mvcGrupo
    $modificar = new mvcGrupo();

    //se manda a llamar el controller para modificar la informacion de un Grupo
    $modificar -> modificarGrupoController();
}
?>

<!--Modal para la confirmacion del editado de un Grupo-->
<div class="modal modal-info fade" id="edit-grupo" style="display: none;">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">×</span></button>
                <h4 class="modal-title">Confirmacion de Editado</h4>
            </div>
            <!--formulario para pedir al usuario su contraseña para confirmar el editado de un
            grupo-->
            <form id="formEdit" role="form" method="post" autocomplete="off" action="index.php?
            section=grupo&action=listado">
                <div class="modal-body">

                    <!--Alert para notificar al usuario que no ha introducido bien su contraseña-->
                    <div class="alert alert-danger alert-dismissible ocultar" id="errorEdit">
                        <button type="button" class="close" onclick="ocultar()">×</button>
                        <h4><i class="icon fa fa-ban"></i>Error</h4>
                        La Contraseña es Incorrecta
                    </div>
```

eliminar.php

En este archivo muestra un modal en la que nos pide nuestra contraseña para eliminar un registro del sistema de la sección en la que nos encontremos, para ello primero verifica si hemos iniciado sesión en el sistema en el sistema.

Después nos muestra un modal con un input en el cual se debe ingresar la contraseña del usuario actual para poder eliminar el registro y enviar el id del registro al controlador de eliminar de la clase y una vez ingresado correctamente la contraseña nos anda a la sección de eliminar de la sección.

En caso de que nos encontremos en la parte de eliminar crea un de la clase que corresponde a la secciona para llamar el método de eliminar de la clase, para así borrar el registro del sistema.

```php
<?php
//verificamos si el usuario ya ha iniciado session
if(!isset($_SESSION["nombre"]))
{
    //si no ha iniciado sesion, lo redireccionara a la seccion publica
    echo "<script>
            window.location.replace('index.php');
          </script>";
}

//verificamos si se debe mandar a llamar el controller para eliminar un grupo del sistema
if(isset($_GET["action"]) && $_GET["action"]=="eliminar")
{
    //se crea un objeto de mvcGrupo
    $eliminar = new mvcGrupo();

    //se manda a llamar el controller para eliminar un Grupo
    $eliminar -> eliminarGrupoController();
}
?>

<!--Modal para la confirmacion del borrado de un grupo-->
<div class="modal modal-info fade" id="del-grupo" style="display: none;">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">×</span></button>
                <h4 class="modal-title">Confirmacion de Borrado</h4>
            </div>
            <!--formulario para pedir al usuario su contraseña para confirmar el borrado del grupo--
            >
            <form id="formDelP" role="form" method="post" autocomplete="off" action="index.php?
            section=grupo&action=eliminar">
                <div class="modal-body">

                    <!--Alert para notificar al usuario que no ha introducido bien su contraseña-->
                    <div class="alert alert-danger alert-dismissible ocultar" id="eliminar">
                        <button type="button" class="close" onclick="ocultar()">×</button>
                        <h4><i class="icon fa fa-ban"></i>Error</h4>
                        La Contraseña es Incorrecta
                    </div>
```

index.php

Este es el archivo principal del sistema en el se incluyen todos los modelos y controladores del sistema, ademas en el se crea un objeto de mvcController para llamar a su método template para hacer funcionar el sistema.

```php
<?php

//incluimos en esta seccion todo lo nesesario para el funcionamiento del sitio
require_once("controllers/controller.php");
require_once("controllers/controllerGrupo.php");
require_once("controllers/controllerAlumno.php");
require_once("controllers/controllerPago.php");
require_once("models/crud.php");
require_once("models/crudGrupo.php");
require_once("models/crudAlumno.php");
require_once("models/crudPago.php");
require_once("models/url.php");

//creamos un objeto de mvcController
$MVC = new mvcController();

//y obtenemos el template
$MVC -> template();
?>
```