UNIVERSIDAD POLITÉCNICA DE VICTORIA

MANUAL TÉCNICO

SISTEMA DE CAI

TECNOLOGÍA Y APLICACIONES WEB

BRIAN ELÍ BECERRA HERNÁNDEZ

ANGELA JUDTIH CARRIZALES PÉREZ

FRANCISCO ISAAC PERALES MORALES

INTRODUCCIÓN

El Centro de Aprendizaje de Idiomas por sus siglas CAI, es una aula donde se llevan a cabo distintas actividades para mejorar las habilidades de escritura, lectura y de comprensión de un idioma.

Los estudiantes de la Universidad Politécnica de Victoria que estén cursando un nivel de inglés tienen que acudir obligatoriamente una hora por semana a CAI, ya que es parte de la evaluación de la materia de inglés.

Lo que se hace actualmente es:

Cada teacher cuenta con uno o más grupos de alumnos a los que les imparte clase.

Se cuentan con listas donde se registran la cantidad de horas que se hace por semana en una unidad determinada. Este registro se hace en un hoja de Google.

Existen 3 teachers que son los encargados de editar estas listas y los demás sólo pueden ver los registros.

Lo que hace el sistema realizado:

Para los usuarios que sean administradores:

Cuentan con un módulo de actividades, donde se registran las actividades que un alumno puede hacer en CAI, las actividades pueden ser agregadas, editadas y eliminadas.

Cuentan con un módulo de grupos, donde previamente debe haber registros de alumnos (también se cuenta con este módulo) donde estos deben estar agregados o deben pertenecer a una carrera. Los grupos pertenecen a un teacher y ahí se van agregando a los alumnos que no cuenten con un grupo.

También se cuenta con un módulo de Unidades, que son las unidades que se darán durante el cuatrimestre, cada una cuenta con una fecha de inicio y una fecha de fin.

Y está el apartado más importante, que es el registro de asistencia de los alumnos cuando van a realizar su hora de CAI. Lo que se hace aquí es ingresar el nombre de un alumno, el cual al ser encontrado te enviará una pantalla con su información y una foto para confirmar que sea el alumno, se agrega a la sesión actual, con esto se quiere decir que la opción para agregar alumnos solo estará 10 minutos abierta después de la hora de inicio en la que se puede hacer CAI. Cada alumno registrado en esta sesión cuenta con una opción para finalizar su hora y si los minutos entre la hora en que se registró y la hora en que se finalizó es menor a 45 minutos no contará como una hora válida de CAI, en caso contrario si se contará. En el momento que vaya a iniciarse otra hora se finaliza la sesión de todos aquellos alumnos que por un motivo u otro no finalizaron su hora individualmente.

Los usuarios pueden filtrar búsquedas, por grupo y por unidad y ver la lista de alumnos que pertenecen a esa búsqueda y al seleccionar un alumno se verán las horas que ha hecho en cada unidad y la actividad que ha ido a realizar.

Para los usuarios normales, es decir, que no son administradores:

Se les muestran los grupos a los que les imparte clase y al seleccionar un alumno se le despliega la información, es decir, las horas de CAI que tiene y qué actividades ha realizado.

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 contiene esta clase es **template**, este controlador incluye en la sección del sistema en que nos encontremos la plantilla con todos los estilos y archivos para el funcionamiento del sistema.

```php
<?php

class mvcController
{
    public function template()
    {
        include "views/template.php";
    }
```

El siguiente controlador es **urlController**, este controlador obtiene mediante el método GET, la sección que debe desplegase 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,

```php
public function urlController()
{
    if(isset($_GET["section"]))
    {
        $section = $_GET["section"];
    }
    else
    {
        $section = "index";
    }

    if(isset($_GET["action"]))
    {
        $action = $_GET["action"];
    }
    else
    {
        $action = "index";
    }

    $url = url::urlModel($section,$action);

    include $url;
}
```

la cual nos retornara el archivo con la sección que nos debe desplegar para lo que debamos realizar en ella.

El siguiente controlador es **loginController**, este controlador obtiene el usuario y la contraseña ingresados en el formulario de inicio de sesión. Esta información se la enviá a loginModel el cual nos retornara el usuario registrado en la base de datos que coincida con el usuario ingresado en el formulario y en caso de que los datos ingresados coincidan con los devueltos por el modelo se inicia sesión el usuario puede iniciar sesión y se lo redirige a la zona de administración del sistema dependiendo de que tipo de usuario que sea.

```
public function loginController()
{

    if(isset($_POST["user"]))
    {

        $data = array("usuario"=>$_POST["user"],
                      "password"=>$_POST["password"]);


        $resp = CRUD::loginModel($data,"usuario");


        if($resp["username"] == $_POST["user"]  && $resp["password"] == $_POST["password"])
        {

            $_SESSION["nombre"] = $resp["nombre"];
            $_SESSION["tipo"] = $resp["tipo"];
            $_SESSION["password"] = $resp["password"];
            $_SESSION["empleado"] = $resp["num_empleado"];


            if($_SESSION["tipo"]=="Administrator")
            {

                echo "<script>
                    window.location.replace('index.php?section=dashboard');
                  </script>";
            }
            else if($_SESSION["tipo"]=="Teacher")
            {

                echo "<script>
                    window.location.replace('index.php?section=groups&action=my-groups');
                  </script>";
            }
        }
    }
}
```

```
    public function dash($tabla)
    {

        $valor = CRUD::dashModel($tabla);


        return $valor[0];
    }
}
?>
```

Por ultimo controlador es **dash**: este controlador recibe de dashModel, información de cuantos registro se tienen almacenados en la tabla que se le mande como parámetro.

Conexion.php

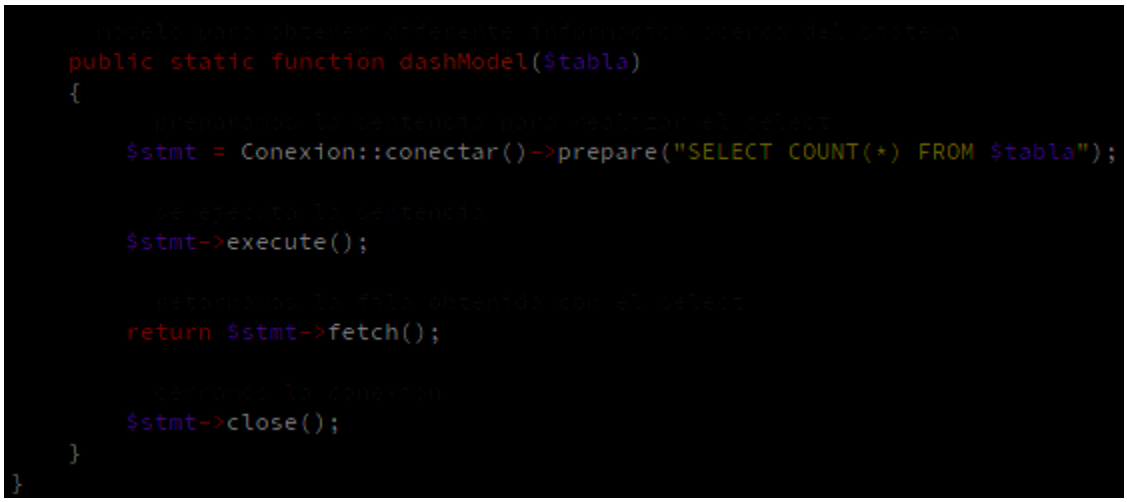
Este archivo contiene la clase **conexión** la cual contiene el modelo **conectar**, esta función le retorna a todos los modelos que la invoquen una conexión a la base de datos para realizar alguna operación de CRUD.

crud.php

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

El primer modelo es **loginModel:** este modelo verifica si existe una coincidencia en la base de datos con los datos ingresados en el login, el cual después de realizar la consulta se la retorna al controlador.

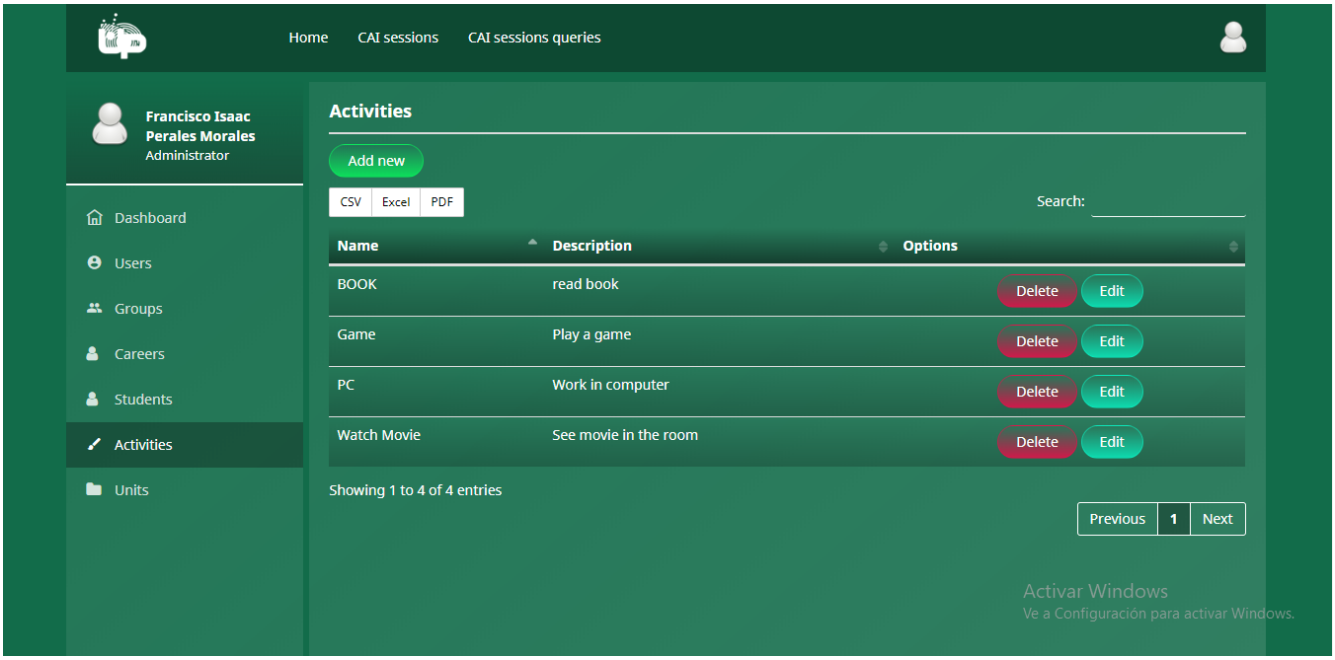
El ultimo modelo es **dashModel:** este modelo cuenta los registros de una tabla, que recibe como parámetro, de la base de datos, el cual después se lo retorna al controlador.



url.php

Este archivo contienen la clase contienen la clase **url,** la cual contiene el modelo de urlModel. Este modelo verifica que archivo debe incluirse para mostrarse en el sistema, dependiendo del section y action que recibe como parámetro, para enviarle el url del archivo al controller para que la incluya.

MÓDULO DE ACTIVIDADES



CONTROLADOR DE ACTIVIDADES (controllerActividad.php)

CONTROLADOR PARA AGREGAR UNA NUEVA ACTIVIDAD

```
function agregarActividadController()
{

    if(isset($_POST["nombre"]))
    {

        $data = array("nombre" => $_POST["nombre"],
                      "descripcion" => $_POST["descripcion"]);


        $resp = CRUDActividad::agregarActividadModel($data,"actividad");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "add";


            echo "<script>
                    window.location.replace('index.php?section=activities&action=list');
                  </script>";
        }
        else
        {

            echo "error";
        }
    }
}
```

En el momento en el que en la vista se de clic en el botón de Agregar se mandará llamar esta función que mediante el método POST obtendrá los valores que se escribieron en el formulario,

esta función los guardará en un arreglo para posteriormente enviarlos al modelo, si en el modelo todo resultó bien el controlador redireccionará a la vista original que mostrará un mensaje de registro exitoso.

CONTROLADOR PARA LISTAR LAS ACTIVIDADES REGISTRADAS

```
function listadoActividadController()
{
    $data = CRUDActividad::listadoActividadModel("actividad");

    foreach($data as $rows => $row)
    {
        echo "<tr class='fondoTabla'>
            <td>".$row["nombre"]."</td>
            <td>".$row["descripcion"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' id='eliminar' data-toggle='modal'
                        data-target='#delete-modal' onclick=idDel('".$row["id actividad"]."
                        ')>Delete</button>
                    <a href='index.php?section=activities&action=list&edit=".$row["id actividad"].
                        "'>
                        <button class='btn btn-rounded btn-custom'>Edit</button>
                    </a>
                </center>
            </td>
        </tr>";
    }
}
```

Esta función se manda llamar en la vista principal de Actividades, muestra las actividades en forma de tabla, cada actividad tiene los datos de nombre y descripción así como las opciones de editar y eliminar.

CONTROLADOR PARA ELIMINAR UNA ACTIVIDAD

```
public function eliminarActividadController()
{

    if(isset($_POST["del"]))
    {

        $data = $_POST["del"];


        $resp = CRUDActividad::eliminarActividadModel($data,"asistencia","actividad");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "delete";


            echo "<script>
                    window.location.replace('index.php?section=activities&action=list');
                  </script>";
        }
    }
}
```

Cuando se da clic en el botón de Eliminar, este ejecuta este controlador que obtiene el id del registro a eliminar, el id es enviado al modelo que será el encargado de eliminarlo de las tablas donde se encuentre registrado en la base de datos, si se eliminó correctamente redireccionará a la página donde mostrará un mensaje de eliminación exitosa.

CONTROLADOR PARA EDITAR UNA ACTIVIDAD

```
public function editarActividadController()
{

    $data = $_GET["edit"];

    $resp = CRUDActividad::editarActividadModel($data,"actividad");

    echo "
                <input type=hidden value=".$resp["id_actividad"]." name='id_actividad'>

                <div class='form-group'>
                    <label class='control-label repairtext'>ID</label>
                    <input type='text' class='form-control' placeholder='ID' value=".$resp["
                        id_actividad"]." readonly>
                </div>

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

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

Esta función a través del Método GET obtiene el id del registro a editar, envía el id al modelo y el modelo regresará los datos del registro y con esto se genera un formulario con los datos del registro.

CONTROLADOR PARA MODIFICAR EL REGISTRO

```
public function modificarActividadController()
{

    if(isset($_POST["nombre"]))
    {

        $data = array("id_actividad" => $_POST["id_actividad"],
                      "nombre" => $_POST["nombre"],
                      "descripcion" => $_POST["descripcion"]);


        $resp = CRUDActividad::modificarActividadModel($data,"actividad");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "edit";


            echo "<script>
                    window.location.replace('index.php?section=activities&action=list');
                  </script>";
        }
    }
}
```

Una vez que se haya generado el formulario con los datos del registro a editar, al dar clic al botón de Modificar se le enviarán los datos al controlador por medio del método POST y este los almacenará en un arreglo que será enviado al modelo para que este se encargue de hacer

el update, si todo salió correctamente el controlador redireccionará a la página del listado y mostrará un mensaje de que el registró se modificó exitosamente.

CONTROLADOR PARA LISTAR LAS ACTIVIDADES EN UN SELECT

```php
public function optionActividadController()
{

    $data = CRUDActividad::optionActividadModel("actividad");


    foreach($data as $rows => $row)
    {

        echo "<option value='".$row["id_actividad"]."'>".$row["nombre"]."</option>";
    }
}
```

Esta función se mandará llamar en cualquier formulario que ocupe seleccionar una actividad, el modelo enviará los datos de todas las actividades que encuentre registradas en la base de datos y el controlador mostrará estas actividades en options de un select.

CRUD DE ACTIVIDADES (crudActividad.php)

MODELO PARA AGREGAR UNA ACTIVIDAD

```php
public static function agregarActividadModel($data,$tabla)
{

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


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


    if($stmt -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


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

Esta función recibe los datos de la actividad a agregar y la tabla donde será agregada, lo primero que hace es preparar la sentencia de inserción, después obtiene los datos y por último ejecuta la sentencia, si todo salió correctamente se retornará un “success” y si no un “fail” que el controlador sabrá que hacer con cada uno de los mensajes.

MODELO PARA LISTAR LAS ACTIVIDADES

```
public static function listadoActividadModel($tabla1)
{
    $stmt = Conexion::conectar() -> prepare("SELECT id actividad, nombre, descripcion FROM $tabla1
        ");

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Esta consulta retornará todos los registros de actividades que se encuentren en la base de datos con los siguientes atributos: id de actividad, el nombre y la descripción.

MODELO PARA ELIMINAR UNA ACTIVIDAD

```
public static function eliminarActividadModel($data,$tabla1,$tabla2)
{
    $stmt1 = Conexion::conectar() -> prepare("DELETE FROM $tabla1 WHERE actividad = :id");

    $stmt1 -> bindParam(":id",$data,PDO::PARAM_STR);



    $stmt2 = Conexion::conectar() -> prepare("DELETE FROM $tabla2 WHERE id actividad = :id");

    $stmt2 -> bindParam(":id",$data,PDO::PARAM_STR);

    if($stmt1 -> execute() && $stmt2 -> execute())
    {
        return "success";
    }
    else
    {
        return "fail";
    }

    $stmt1 -> close();
    $stmt2 -> close();
}
```

Lo que recibe esta función es el id de la actividad a eliminar y las tablas de donde hay que eliminar el registro, primero se obtienen los id de las asistencias que tienen el id de la actividad a eliminar, es decir, se obtienen los registros que tengan como clave foránea el id de la actividad, después se obtiene el id de la actividad a eliminar de la tabla de actividades y se ejecutan ambas sentencias y retornan “success” si todo salió bien y “fail” se algo salió mal.

MODELO PARA EDITAR UNA ACTIVIDAD

```
public static function editarActividadModel($data,$tabla1)
{

    $stmt = Conexion::conectar()->prepare("SELECT id actividad, nombre, descripcion FROM $tabla1 WHERE
        id actividad = :id");


    $stmt->bindParam(":id",$data, PDO::PARAM_STR);


    $stmt->execute();


    return $stmt->fetch();


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

Esta función retorna el id de la actividad, el nombre y la descripción del registro que se solicitó .

MODELO PARA MODIFICAR UNA ACTIVIDAD

```
public static function modificarActividadModel($data,$tabla)
{

    $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre, descripcion = :descripcion
        WHERE id actividad = :id");


    $stmt -> bindParam(":id", $data["id actividad"], PDO::PARAM_INT);
    $stmt -> bindParam(":nombre", $data["nombre"], PDO::PARAM_STR);
    $stmt -> bindParam(":descripcion", $data["descripcion"], PDO::PARAM_STR);


    if($stmt -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


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

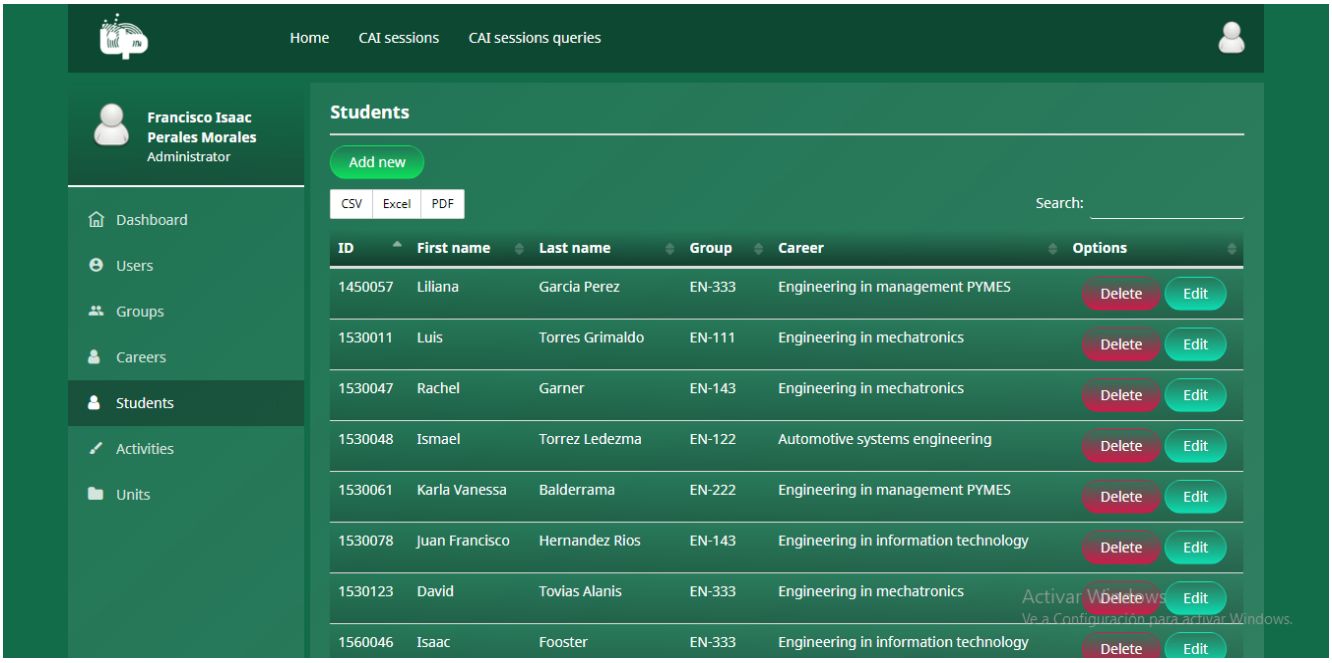
Esta función recibe los datos del registro a modificar y la tabla donde se hará el update, se prepara la sentencia de actualización, se obtienen los datos que requiere la sentencia y se ejecuta, retornando “success” si se modificó correctamente y “fail” en caso contrario.

MODELO PARA OBTENER LOS DATOS DE LAS ACTIVIDADES

```php
public static function optionActividadModel($tabla1)
{
    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla1");

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Retorna todos los datos de todas las actividades registradas en la base de datos.

MÓDULO DE ALUMNOS



CONTROLADOR DE ALUMNOS (controllerAlumno.php)

CONTROLADOR PARA AGREGAR UN ALUMNO

```php
<?php

class mvcAlumno
{

    function agregarAlumnoController()
    {

        if(isset($_POST["matricula"]))
        {

            $data = array("matricula" => $_POST["matricula"],
                          "nombre" => $_POST["nombre"],
                          "apellido" => $_POST["apellido"],
                          "carrera" => $_POST["carrera"],
                          "img" => "views/media/img/noimg.png");


            if(!empty($_FILES["img"]["name"]))
            {

                $type = $_FILES["img"]["type"];


                $size = $_FILES["img"]["size"];


                $name = $_FILES["img"]["name"];


                $tmp = $_FILES["img"]["tmp_name"];


                $date = getdate();


                if($type == "image/jpeg" || $type == "image/png")
                {


                    if($size < 5000000)
                    {
```

Este controlador se encarga de recibir la información y la imagen de un nuevo alumno a registrar en el sistema. Una vez recibida la información y la imagen, la almacena en un array y la enviá al modelo para que sea registrada en la base datos.

CONTROLADOR PARA LISTAR A LOS ALUMNOS

```
function listadoAlumnoController()
{

    $data = CRUDAlumno::listadoAlumnoModel("alumno","grupo","carrera");


    foreach($data as $rows => $row)
    {

        echo "<tr class='fondoTabla'>
            <td>".$row["matricula"]."</td>
            <td>".$row["nombre"]."</td>
            <td>".$row["apellido"]."</td>
            <td>".$row["grupo"]."</td>
            <td>".$row["carrera"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' id='eliminar' data-toggle='modal'
                        data-target='#delete-modal' onclick=idDel('".$row["matricula"]."')>Delete</button>
                    <a href='index.php?section=students&action=list&edit=".$row["matricula"]."'>
                        <button class='btn btn-rounded btn-custom'>Edit</button>
                    </a>
                </center>
            </td>
        </tr>";
    }
}
```

Esta función se encarga de mostrar en forma de tabla todos los registros de alumnos que el modelo retorne, se muestra la matricula, el nombre, apellido, grupo y carrera además de que cada registro tiene las opciones de eliminar y editar.

CONTROLADOR PARA ELIMINAR UN ALUMNO

```
public function eliminarAlumnoController()
{

    if(isset($_POST["del"]))
    {

        $data = $_POST["del"];


        $resp = CRUDAlumno::eliminarAlumnoModel($data,"asistencia","alumno");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "delete";


            echo "<script>
                    window.location.replace('index.php?section=students&action=list');
                  </script>";
        }
    }
}
```

Para eliminar un alumno, esta función recibe el id del registro a eliminar, lo envía al modelo y si el delete fue exitoso es redireccionará a la lista de alumnos y mostrará un mensaje que dirá que el registro fue eliminado exitosamente.

CONTROLADOR PARA EDITAR UN ALUMNO

```
public function editarAlumnoController()
{

    $data = $_GET["edit"];


    $resp = CRUDAlumno::editarAlumnoModel($data,"alumno");


    echo "
                <input type=hidden value=".$resp["matricula"]." name='matricula'>

                 <div class='form-group'>
                    <label class='control-label repairtext'>ID</label>
                    <input type='text' class='form-control' placeholder='Code' readonly value=".$resp["
                        matricula"].">
                </div>

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

                <div class='form-group'>
                    <label class='control-label repairtext'>Last name</label>
                    <input type='text' name='apellido' class='form-control' placeholder='Code' value='".$resp
                        ["apellido"]."' required>
                </div>";

    echo "
                 <div class='form-group'>
                     <label class='control-label repairtext'>Career</label>
                     <select style='width:100%;' class='form-control select2' id='career' name='carrera'
                         required>
                         <option value=''></option>";


                         $option = new mvcCarrera();


                         $option -> optionCarreraController();

     echo "          </select>
                 </div>

                 <div class='form-group'>
                     <label class='control-label repairtext'>Group</label>
                     <select style='width:100%;' class='form-control select2' id='grupo' name='grupo'>
                         <option value=''></option>";

                         $option = new mvcGrupo();


                         $option -> optionGrupoController();
      echo "          </select>
                 </div>";
```

```
    echo "<script>
            var career = document.getElementById('career');
            var grupo = document.getElementById('grupo');

            for(var i = 1; i < career.options.length; i++)
            {
                if(career.options[i].value ==''.$resp["carrera"].'')
                {
                    career.selectedIndex = i;
                }
            }

            for(var i = 1; i < grupo.options.length; i++)
            {
                if(grupo.options[i].value ==''.$resp["grupo"].'')
                {
                    grupo.selectedIndex = i;
                }
            }
        </script>";

}
```

Esta función se encarga de mostrar un formulario con los datos del alumno a editar, aquí se mandan a llamar otros controladores que listan las carreras y grupos registrados en la base de datos en options dentro de un select, por medio de javaScript se hace funcionar el select2, que se encarga de buscar dentro del select.

CONTROLADOR PARA MODIFICAR UN ALUMNO

```
public function modificarAlumnoController()
{

    if(isset($_POST["matricula"]))
    {

        $data = array("matricula" => $_POST["matricula"],
                      "nombre" => $_POST["nombre"],
                      "apellido" => $_POST["apellido"],
                      "carrera" => $_POST["carrera"],
                      "grupo" => $_POST["grupo"]);


        $resp = CRUDAlumno::modificarAlumnoModel($data,"alumno");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "edit";


            echo "<script>
                    window.location.replace('index.php?section=students&action=list');
                </script>";
        }
    }
}
```

Esta función almacena los datos del alumno en un arreglo que envía al modelo para que este se encargue de hacer el update, el modelo retornará success si todo salió correctamente y el controlador redireccionará a la vista de listado de alumnos y mostrará un mensaje que dirá que el registró se actualizó correctamente.

CONTROLADOR PARA MOSTRAR LOS ALUMNOS EN UN SELECT

```
public function optionTodosAlumnosController()
{

    $data = CRUDAlumno::optionTodosAlumnosModel("alumno");


    foreach($data as $rows => $row)
    {

        echo "<option value='".$row["matricula"]."'>".$row["nombre"]." ".$row["apellido"]."</option>";
    }
}
```

El modelo retornará todos los alumnos y esta función se encargará de mostrar en options el nombre y apellido del alumno además como valor oculto su matrícula.

CONTROLADOR PARA MOSTRAR LOS ALUMNOS SIN GRUPO EN UN SELECT

```
public function optionAlumnoController()
{

    $data = CRUDAlumno::optionAlumnoModel("alumno");


    foreach($data as $rows => $row)
    {

        echo "<option class='repairtext'
        value=".$row["matricula"].">".$row["nombre"]." ".$row["apellido"]."</option>";
    }
}
```

Este controlador recibe del modelo la infromacion de todos los alumnos que no estén en un grupo. Una vez que reciba la información, muestra la información de los alumnos sin grupo en un select.

CONTROLADOR PARA AGREGAR ALUMNOS A UN GRUPO

```php
function agregarAlumnoGrupoController()
{

    if(isset($_POST["matricula"]))
    {

        $data = array("matricula" => $_POST["matricula"],
                      "grupo" => $_POST["grupo"]);


        $resp = CRUDAlumno::agregarAlumnoGrupoModel($data,"alumno");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "add";


            echo "<script>
                    window.location.replace('index.php?
                    section=groups&action=students&group=".$data["grupo"]."');
                  </script>";
        }
        else
        {

            echo "error";
        }
    }
}
```

Este controlador se encarga de recibir el id de un alumno y el id del grupo al cual el alumno va a ingresar. Una vez recibida la información la almacena en un array y la enviá al modelo para que registre al alumno en el grupo.

CONTROLADOR PARA LISTAR LOS ALUMNOS DE UN GRUPO

```php
function listadoAlumnoGrupoController()
{
    $data = CRUDAlumno::listadoAlumnoGrupoModel($_GET["group"],"alumno","carrera");

    foreach($data as $rows => $row)
    {
        echo "<tr class='fondoTabla'>
            <td>".$row["matricula"]."</td>
            <td>".$row["nombre"]."</td>
            <td>".$row["apellido"]."</td>
            <td>".$row["carrera"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' id='eliminar' data-
                    toggle='modal' data-target='#delete-modal'
                    onclick=idDel('".$row["matricula"]."')>Delete</button>
                </center>
            </td>
        </tr>";
    }
}
```

Este controlador recibe la información de todos los alumnos que estén registrados en un determinado grupo. Un vez recibida la información, la muestra en una tabla.

CONTROLADOR PARA ELIMINAR UN ALUMNO DE UN GRUPO

```php
public function eliminarAlumnoGrupoController()
{
    if(isset($_POST["del"]))
    {
        $data = array ("matricula" => $_POST["del"],
                       "grupo" => $_POST["grupo"]);

        $resp = CRUDAlumno::eliminarAlumnoGrupoModel($data,"alumno");

        if($resp == "success")
        {
            $_SESSION["mensaje"] = "delete";

            echo "<script>
                    window.location.replace('index.php?
                    section=groups&action=students&group=".$data["grupo"]."');
                  </script>";
        }
    }
}
```

Este controlador recibe del sistema el id del alumno al cual se eliminara del grupo al que pertenece. Una vez recibido el id se lo manda al modelo para que borre al alumno del grupo al que pertenece.

CONTROLADOR PARA LISTAR LOS ALUMNOS CON GRUPO EN UN SELECT

```php
public function optionAlumnosController()
{

    $data = CRUDAlumno::optionAlumnosModel("alumno");


    foreach($data as $rows => $row)
    {

        echo "<option value='".$row["matricula"]."'>".$row["nombre"]." ".$row["apellido"]."</option>";
    }
}
```

El modelo retornará los alumnos con grupo y esta función se encargará de mostrarlos en options dentro de un select.

CRUD DE ALUMNOS (crudAlumno.php)

MODELO PARA AGREGAR UN ALUMNO

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


class CRUDAlumno
{

    public static function agregarAlumnoModel($data,$tabla)
    {

        $stmt = Conexion::conectar() -> prepare("INSERT INTO $tabla
        (matricula,nombre,apellido,carrera,img) VALUES
        (:matricula,:nombre,:apellido,:carrera,:img)");


        $stmt -> bindParam(":matricula",$data["matricula"],PDO::PARAM_INT);
        $stmt -> bindParam(":nombre",$data["nombre"],PDO::PARAM_STR);
        $stmt -> bindParam(":apellido",$data["apellido"],PDO::PARAM_STR);
        $stmt -> bindParam(":carrera",$data["carrera"],PDO::PARAM_STR);
        $stmt -> bindParam(":img",$data["img"],PDO::PARAM_STR);


        if($stmt -> execute())
        {

            return "success";
        }
        else
        {

            return "fail";
        }


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

Este modelo recibe del controlador, la información y la imagen de un nuevo alumno, el cual mediante este modelo se registrara en la base de datos.

MODELO PARA LISTAR A LOS ALUMNOS

```
public static function listadoAlumnoModel($tabla1,$tabla2,$tabla3)
{

    $stmt = Conexion::conectar() -> prepare("SELECT a.matricula as matricula , a.nombre as nombre, a.apellido
        as apellido, a.grupo as grupo, c.nombre as carrera
                                            FROM $tabla1 as a
                                            JOIN $tabla3 as c on c.siglas = a.carrera");


    $stmt -> execute();


    return $stmt -> fetchAll();


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

A través de esta sentencia se obtiene el nombre de la carrera y grupo y no sus id, ya que no nos interesa observar el id sino el nombre, se ejecuta la sentencia y se retornan los valores devueltos por la consulta.

MODELO PARA ELIMINAR UN ALUMNO

```
public static function eliminarAlumnoModel($data,$tabla1,$tabla2)
{

    $stmt1 = Conexion::conectar() -> prepare("DELETE FROM $tabla1 WHERE alumno = :id");


    $stmt1 -> bindParam(":id",$data,PDO::PARAM_INT);



    $stmt2 = Conexion::conectar() -> prepare("DELETE FROM $tabla2 WHERE matricula = :id");


    $stmt2 -> bindParam(":id",$data,PDO::PARAM_INT);


    if($stmt1 -> execute() && $stmt2 -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


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

Primero se obtienen los id de las asistencias donde ese alumno se encuentra, es decir, se obtienen los registros que tengas como llave foránea el id del alumno a eliminar, después se realiza la sentencia para eliminar el alumno de la tablas alumnos, se ejecutan ambas sentencias y se retorna “success” si todo salió bien y “fail” en caso contrario.

MODELO PARA EDITAR UN ALUMNO

```php
public static function editarAlumnoModel($data,$tabla)
{

    $stmt = Conexion::conectar()->prepare("SELECT matricula, nombre, apellido, carrera, grupo
                                           FROM $tabla
                                           WHERE matricula = :id");


    $stmt->bindParam(":id",$data, PDO::PARAM_INT);


    $stmt->execute();


    return $stmt->fetch();


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

Esta función retornará los datos del alumno a editar.

MODELO PARA MODIFICAR A UN ALUMNO

```php
public static function modificarAlumnoModel($data,$tabla)
{

    $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre, apellido = :apellido, carrera
        = :carrera, grupo = :grupo WHERE matricula = :id");

    if(empty($data["grupo"]))
    {
        $data["grupo"] = null;
    }


    $stmt -> bindParam(":id", $data["matricula"], PDO::PARAM_INT);
    $stmt -> bindParam(":nombre", $data["nombre"], PDO::PARAM_STR);
    $stmt -> bindParam(":apellido", $data["apellido"], PDO::PARAM_STR);
    $stmt -> bindParam(":carrera", $data["carrera"], PDO::PARAM_STR);
    $stmt -> bindParam(":grupo", $data["grupo"], PDO::PARAM_STR);


    if($stmt -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


    $stmt->close();
```

Se prepara la sentencia de actualización, se obtienen los datos del registro a actualizar y se ejecuta el update, retornando “success” si todo salió bien y “fail” en caso contrario.

MODELO PARA MOSTRAR TODOS LOS ALUMNOS EN UN SELECT

```
public static function optionTodosAlumnosModel($tabla)
{
    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla");

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Se retornan todos los alumnos registrados en la base de datos.

MODELO PARA MOSTRAR TODOS LOS ALUMNOS SIN GRUPO EN UN SELECT

```
public static function optionAlumnoModel($tabla)
{
    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla WHERE grupo is
    NULL");

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Este modelo obtiene de la base de datos, la información de los alumnos que no tienen un grupo, una vez que obtiene la información la retorna al controlador para mostrarla en un select.

MODELO PARA AGREAR UN ALUMNO A UN GRUPO

```
public static function agregarAlumnoGrupoModel($data,$tabla)
{

    $stmt = Conexion::conectar() -> prepare("UPDATE $tabla SET grupo = :grupo WHERE
    matricula = :matricula");

    $stmt -> bindParam(":matricula",$data["matricula"],PDO::PARAM_INT);
    $stmt -> bindParam(":grupo",$data["grupo"],PDO::PARAM_STR);

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

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

Este modelo recibe del controlador el id de un alumno y el id de un grupo al cual el alumno va a ingresar. Una vez obtenido esta información, en modelo modifica el registro del alumnos para que este relacionado con el grupo que recibió el modelo.

MODELO PARA LISTAR LOS ALUMNOS DE UN GRUPO

```
public static function listadoAlumnoGrupoModel($data,$tabla1,$tabla2)
{

    $stmt = Conexion::conectar() -> prepare("SELECT a.nombre, a.apellido, a.matricula,
    c.nombre as carrera
                                            FROM $tabla1 as a
                                            JOIN $tabla2 as c on c.siglas = a.carrera
                                            WHERE grupo = :grupo");

    $stmt -> bindParam(":grupo",$data,PDO::PARAM_STR);

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Este modelo obtiene la información, de la base de datos, de todos los alumnos que pertenezcan a un determinado grupo. Una vez que obtiene la información, la retorna al controlador para ser mostrada.

MODELO PARA ELIMINAR UN ALUMNO DE UN GRUPO

```php
public static function eliminarAlumnoGrupoModel($data,$tabla)
{
    $stmt = Conexion::conectar() -> prepare("UPDATE $tabla SET grupo = NULL WHERE
    matricula = :id");

    $stmt -> bindParam(":id",$data["matricula"],PDO::PARAM_INT);

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

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

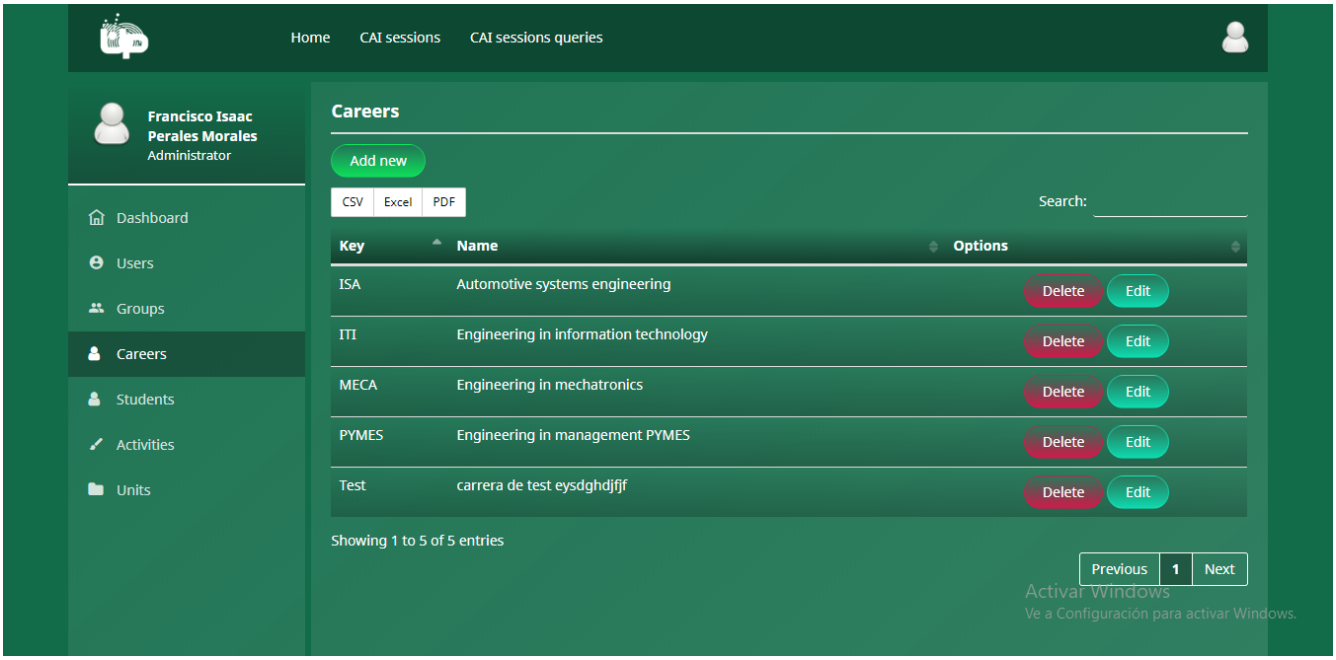
Este modelo recibe del controlador el id del alumno al cual se eliminara del grupo al que pertenece. Una vez obtenido el id, el modelo actualiza el registro del alumno para que ya no este relacionado con ese grupo.

MODELO PARA MOSTRAR LOS ALUMNOS CON GRUPO EN UN SELECT

```php
public static function optionAlumnosModel($tabla)
{
    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla WHERE grupo IS NOT NULL");

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Retorna todos alumnos que tienen un grupo asignado.

MÓDULO DE CARRERAS



CONTROLADOR DE CARRERAS (controllerCarrera.php)

CONTROLADOR PARA AGREGAR UNA CARRERA

```
function agregarCarreraController()
{

    if(isset($_POST["siglas"]))
    {

        $data = array("siglas" => $_POST["siglas"],
                      "nombre" => $_POST["nombre"]);


        $resp = CRUDCarrera::agregarCarreraModel($data,"carrera");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "add";


            echo "<script>
                    window.location.replace('index.php?section=careers&action=list');
                  </script>";
        }
        else
        {

            echo "error";
        }
    }
}
```

Por medio del método POST obtiene los datos del formulario, los agrega a un arreglo que se envía al modelo, si se registró correctamente se redireccionará a la vista de listado de carreras y mostrará un mensaje de que se registró exitosamente.

CONTROLADOR PARA LISTAR LAS CARRERAS

```
function listadoCarreraController()
{
    $data = CRUDCarrera::listadoCarreraModel("carrera");

    foreach($data as $rows => $row)
    {
        echo "<tr class='fondoTabla'>
            <td>".$row["siglas"]."</td>
            <td>".$row["nombre"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' id='eliminar' data-toggle='modal'
                        data-target='#delete-modal' onclick=idDel('".$row["siglas"]."')>Delete</button>
                    <a href='index.php?section=careers&action=list&edit=".$row["siglas"]."'>
                        <button class='btn btn-rounded btn-custom'>Edit</button>
                    </a>
                </center>
            </td>
        </tr>";
    }
}
```

Obtiene los datos de todas las carreras registradas y las muestra en una tabla cada registro con dos opciones, Eliminar y Editar.

CONTROLADOR PARA ELIMINAR UNA CARRERA

```
public function eliminarCarreraController()
{

    if(isset($ POST["del"]))
    {

        $data = $ POST["del"];


        $resp = CRUDCarrera::eliminarCarreraModel($data,"asistencia","alumno","carrera");


        if($resp == "success")
        {

            $ SESSION["mensaje"] = "delete";


            echo "<script>
                    window.location.replace('index.php?section=careers&action=list');
                  </script>";
        }
    }
}
```

Se obtiene el id de la carrera a eliminar, el id es enviado al modelo junto con las tablas donde aparece la carrera y si el delete fue exitoso se redireccionará a la vista de listado de carreras y mostrará un mensaje donde dirá que el delete fue exitoso.

CONTROLADOR PARA EDITAR UNA CARRERA

```
public function editarCarreraController()
{

    $data = $ GET["edit"];


    $resp = CRUDCarrera::editarCarreraModel($data,"carrera");


    echo "
              <input type=hidden value=".$resp["siglas"]." name='siglas'>

              <div class='form-group'>
                  <label class='control-label repairtext'>Code</label>
                  <input type='text' class='form-control' placeholder='Code' value=".$resp["siglas"]."
                     readonly>
              </div>

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

Se obtiene el id de la carrera a editar por medio del método GET y se muestra un formulario con los datos de la carrera retornados por el modelo.

CONTROLADOR PARA MODIFICAR UNA CARRERA

```
public function modificarCarreraController()
{

    if(isset($ POST["siglas"]))
    {

        $data = array("siglas" => $ POST["siglas"],
                      "nombre" => $ POST["nombre"]);


        $resp = CRUDCarrera::modificarCarreraModel($data,"carrera");


        if($resp == "success")
        {

            $ SESSION["mensaje"] = "edit";


            echo "<script>
                    window.location.replace('index.php?section=careers&action=list');
                </script>";
        }
    }
}
```

Se obtienen los datos de la carrera del formulario de editar, se envían al modelo y si el update fue correcto se redireccionará a la vista de listado de carreras y mostrará un mensaje que dirá que la carrera fue actualizada exitosamente.

CONTROLADOR PARA MOSTRAR LAS CARRERAS EN UN SELECT

```php
public function optionCarreraController()
{

    $data = CRUDCarrera::optionCarreraModel("carrera");


    foreach($data as $rows => $row)
    {

        echo "<option value='".$row["siglas"]."'>".$row["nombre"]."</option>";
    }
}
```

A través del modelo obtendrá todas las carreras y las mostrará en options dentro de un select.

CRUD DE CARRERAS (crudCarrera.php)

MODELO PARA AGREGAR UNA CARRERA

```php
public static function agregarCarreraModel($data,$tabla)
{

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


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


    if($stmt -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


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

Se prepara la sentencia de insercción, se obtienen los datos que la sentencia necesita y se ejecuta, retornando “success” si se hizo la insercción correctamente y “fail” en caso contrario.

MODELO PARA LISTAR LAS CARRERAS

```php
public static function listadoCarreraModel($tabla)
{

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


    $stmt -> execute();


    return $stmt -> fetchAll();


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

Este modelo retornará todas las carreras registradas en la base de datos.

MODELO PARA ELIMINAR UNA CARRERA

```php
public static function eliminarCarreraModel($data,$tabla1,$tabla2,$tabla3)
{

    $stmt = Conexion::conectar() -> prepare("SELECT matricula FROM $tabla2 WHERE carrera = :id");


    $stmt -> bindParam(":id",$data,PDO::PARAM_STR);


    $stmt -> execute();


    $students = $stmt -> fetchAll();


    foreach($students as $rows => $row)
    {

        $resp = CRUDAlumno::eliminarAlumnoModel($row["matricula"],$tabla1,$tabla2);
    }



    $stmtl = Conexion::conectar() -> prepare("DELETE FROM $tabla3 WHERE siglas = :id");


    $stmtl -> bindParam(":id",$data,PDO::PARAM_STR);


    if($stmtl -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


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

Al eliminar una carrera se eliminarán los alumnos que están asociados a esa carrera, es por ello que se obtienen las matriculas de los alumnos que pertenecen a esa carrera y se eliminan y después se realiza la sentencia de delete de esa carrera y se ejecuta, si todo salió bien se retorna “success” y “fail” en caso contrario.

MODELO PARA EDITAR UNA CARRERA

```php
public static function editarCarreraModel($data,$tabla1)
{

    $stmt = Conexion::conectar()->prepare("SELECT siglas, nombre FROM $tabla1 WHERE siglas = :id");


    $stmt->bindParam(":id",$data, PDO::PARAM_STR);


    $stmt->execute();


    return $stmt->fetch();


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

Se retornarán los datos de la carrera que se envío.

MODELO PARA MODIFICAR UNA CARRERA

```php
public static function modificarCarreraModel($data,$tabla)
{

    $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre WHERE siglas = :id");


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


    if($stmt -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


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

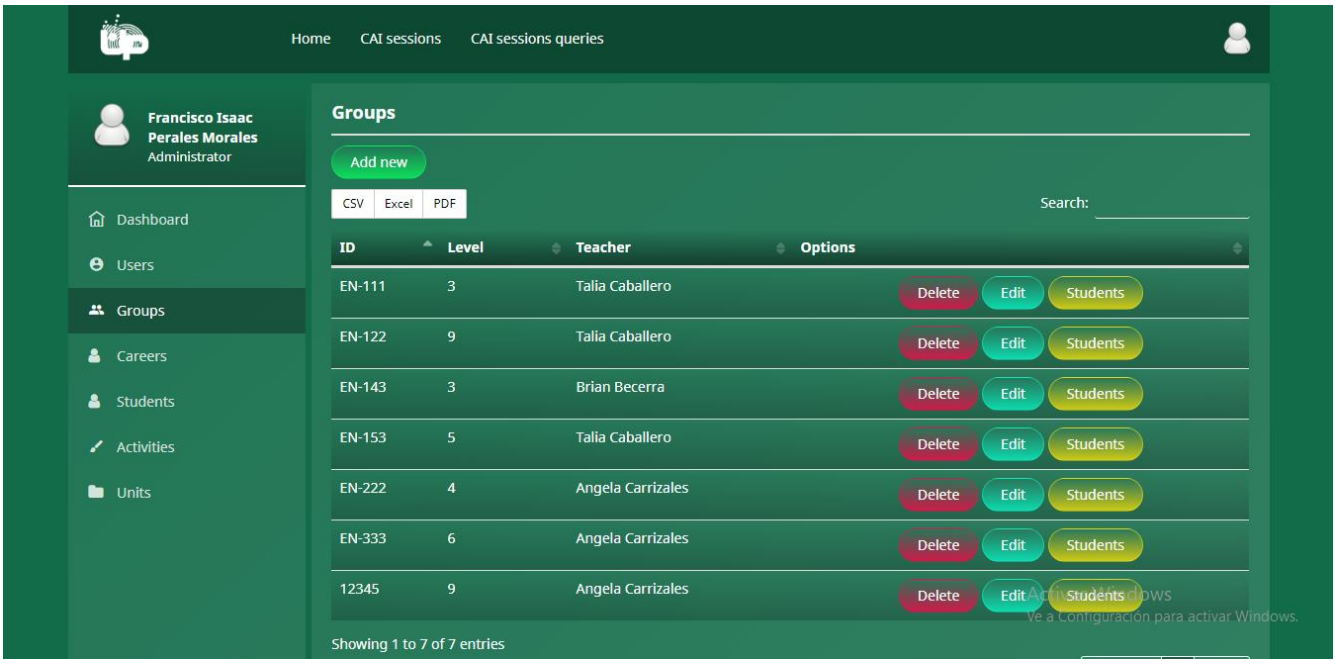
Se prepara la sentencia para modificar el registro en la base de datos, se obtienen los datos que necesita la sentencia y se ejecuta, si el update fue correcto se retorna “success” y “fail” en caso contrario.

MODELO PARA MOSTRAR LAS CARRERAS EN UN SELECT

```php
public static function optionCarreraModel($tabla1)
{

    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla1");


    $stmt -> execute();


    return $stmt -> fetchAll();


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

Retorna los datos de todas las carreras para luego ser mostradas en options dentro de un select.

MÓDULO DE GRUPOS



CONTROLADOR DE GRUPOS (controllerGrupo.php)

CONTROLADOR PARA AGREGAR UN GRUPO

```php
<?php

class mvcGrupo
{

    function agregarGrupoController()
    {

        if(isset($_POST["codigo"]))
        {

            $datos = array("codigo" => $_POST["codigo"],
                           "nivel" => $_POST["nivel"],
                           "teacher" => $_POST["teacher"]);


            $resp = CRUDGrupo::agregarGrupoModel($datos,"grupo");
```

```
            if($resp == "success")
            {

                $_SESSION["mensaje"] = "add";


                echo "<script>
                        window.location.replace('index.php?section=groups&action=list');
                      </script>";
            }
            else
            {

                echo "error";
            }
        }
    }
```

Este controlador se encarga de recibir la información de un nuevo grupo a registrar en el sistema. Una vez recibida la información la almacena en un array y la enviá al modelo para que sea registrada en la base datos.

CONTROLADOR PARA LISTAR LOS GRUPOS

```
function listadoGrupoController()
{


    $data = CRUDGrupo::listadoGrupoModel("grupo","teacher","usuario");


    foreach($data as $rows => $row)
    {

        echo "<tr class='fondoTabla'>
            <td>".$row["codigo"]."</td>
            <td>".$row["nivel"]."</td>
            <td>".$row["teacher"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' id='eliminar' data-
                    toggle='modal' data-target='#delete-modal'
                    onclick=idDel('".$row["codigo"]."')>Delete</button>
                    <a href='index.php?
                    section=groups&action=list&edit=".$row["codigo"]."'>
                        <button class='btn btn-rounded btn-custom'>Edit</button>
                    </a>
                    <a href='index.php?
                    section=groups&action=students&group=".$row["codigo"]."'>
                        <button class='btn btn-rounded btn-warning'>Students</button>
                    </a>
                </center>
            </td>
        </tr>";
    }
}
```

Este controlador recibe del modelo la información de todos los grupos registrados en el sistema, para así imprimir su información en pantalla para ser mostrada mediante una tabla.

CONTROLADOR PARA ELIMINAR UN GRUPO

```php
public function eliminarGrupoController()
{
    if(isset($_POST["del"]))
    {
        $data = $_POST["del"];

        $resp = CRUDGrupo::eliminarGrupoModel($data,"alumno","grupo");

        if($resp == "success")
        {
            $_SESSION["mensaje"] = "delete";

            echo "<script>
                    window.location.replace('index.php?section=groups&action=list');
                  </script>";
        }
    }
}
```

Este controlador recibe del el sistema el id del grupo a eliminar de la base de datos. Una vez recibido el id se lo enviá al modelo para su borrado en la base de datos.

CONTROLADOR PARA EDITAR UN GRUPO

```php
public function editarGrupoController()
{
    $data = $_GET["edit"];

    $resp = CRUDGrupo::editarGrupoModel($data,"grupo","teacher","usuario");

    echo "
                <input type=hidden value=".$resp["codigo"]." name='id'>

                 <div class='form-group'>
                    <label class='control-label repairtext'>Code</label>
                    <input type='text' class='form-control' placeholder='Code'
                    value=".$resp["codigo"]." readonly>
                </div>

                <div class='form-group'>
                    <label class='control-label repairtext'>Level</label>
                    <select style='width:100%;' class='form-control select2'
                    id='level' name='nivel' required>
                        <option value=''></option>";
                        for($i = 1; $i <= 9; $i++)
                        {
                            echo "<option value=".$i.">Level ".$i."</option>";
```

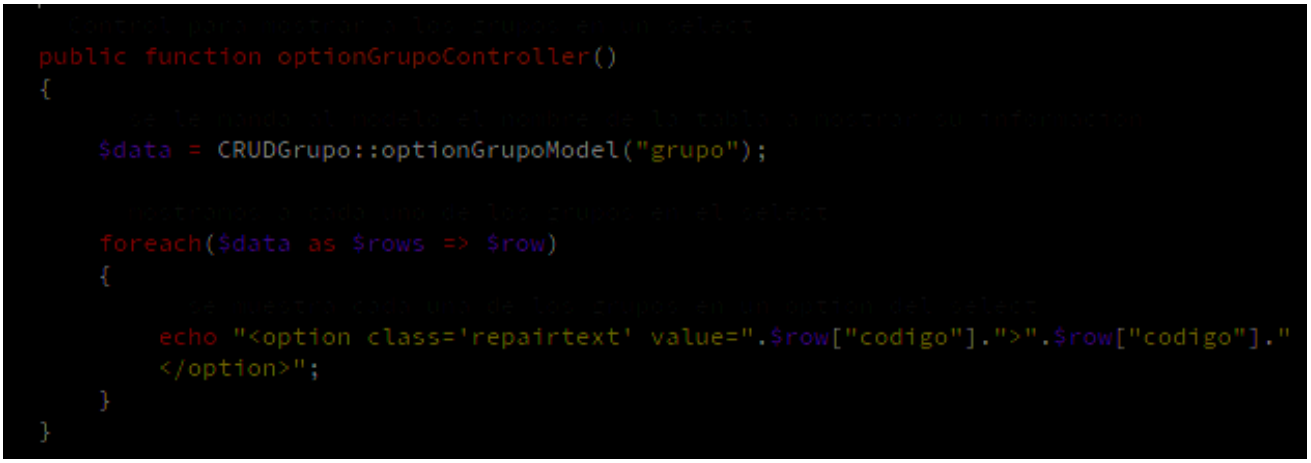
Este controlador recibe del modelo, la información de un grupo en específico, para mostrar su información en un formulario para ser editada en caso de ser necesario

CONTROLADOR PARA MODIFICAR UN GRUPO

```
public function modificarGrupoController()
{
    if(isset($_POST["id"]))
    {
        $data = array("id" => $_POST["id"],
                      "nivel" => $_POST["nivel"],
                      "teacher" => $_POST["teacher"]);

        $resp = CRUDGrupo::modificarGrupoModel($data,"grupo");

        if($resp == "success")
        {
            $_SESSION["mensaje"] = "edit";

            echo "<script>
                    window.location.replace('index.php?section=groups&action=list');
                </script>";
        }
    }
}
```

Este controlador recibe la información modificada de un grupo, después esta información se la enviá al modelo para actualizar la información del grupo en la base de datos.

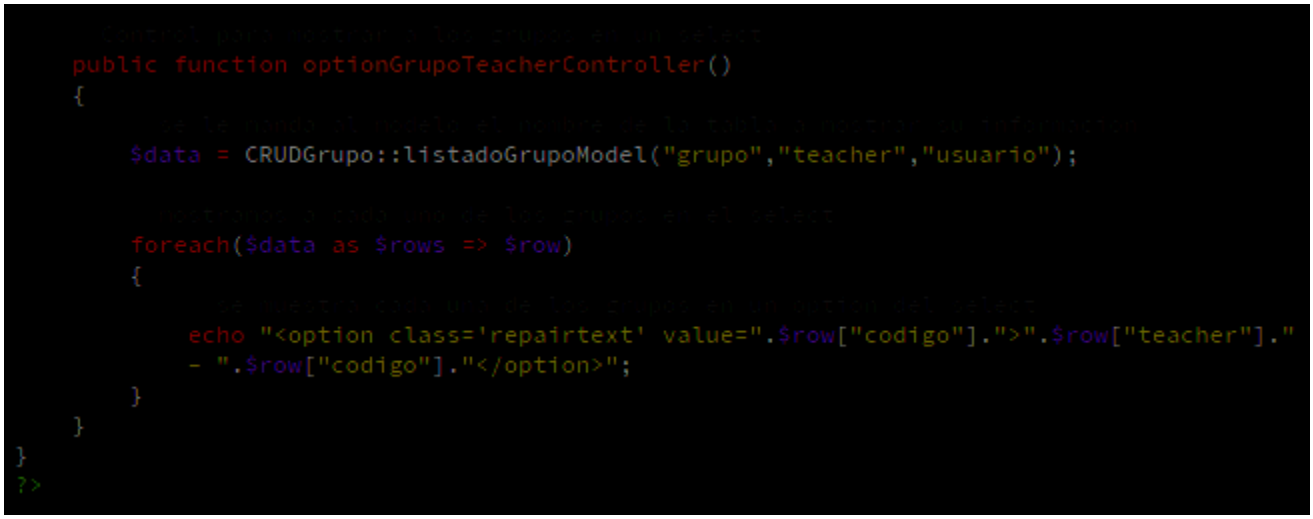
CONTROLADOR PARA MOSTRAR LOS GRUPOS EN UN SELECT



```
public function optionGrupoController()
{
    $data = CRUDGrupo::optionGrupoModel("grupo");

    foreach($data as $rows => $row)
    {
        echo "<option class='repairtext' value=".$row["codigo"].">".$row["codigo"]."
        </option>";
    }
}
```

Este controlador recibe la información de los grupos de la base de datos para después mostrar su información mediante un select.

CONTROLADOR PARA MOSTRAR LOS GRUPOS DE UN TEACHER EN UN SELECT

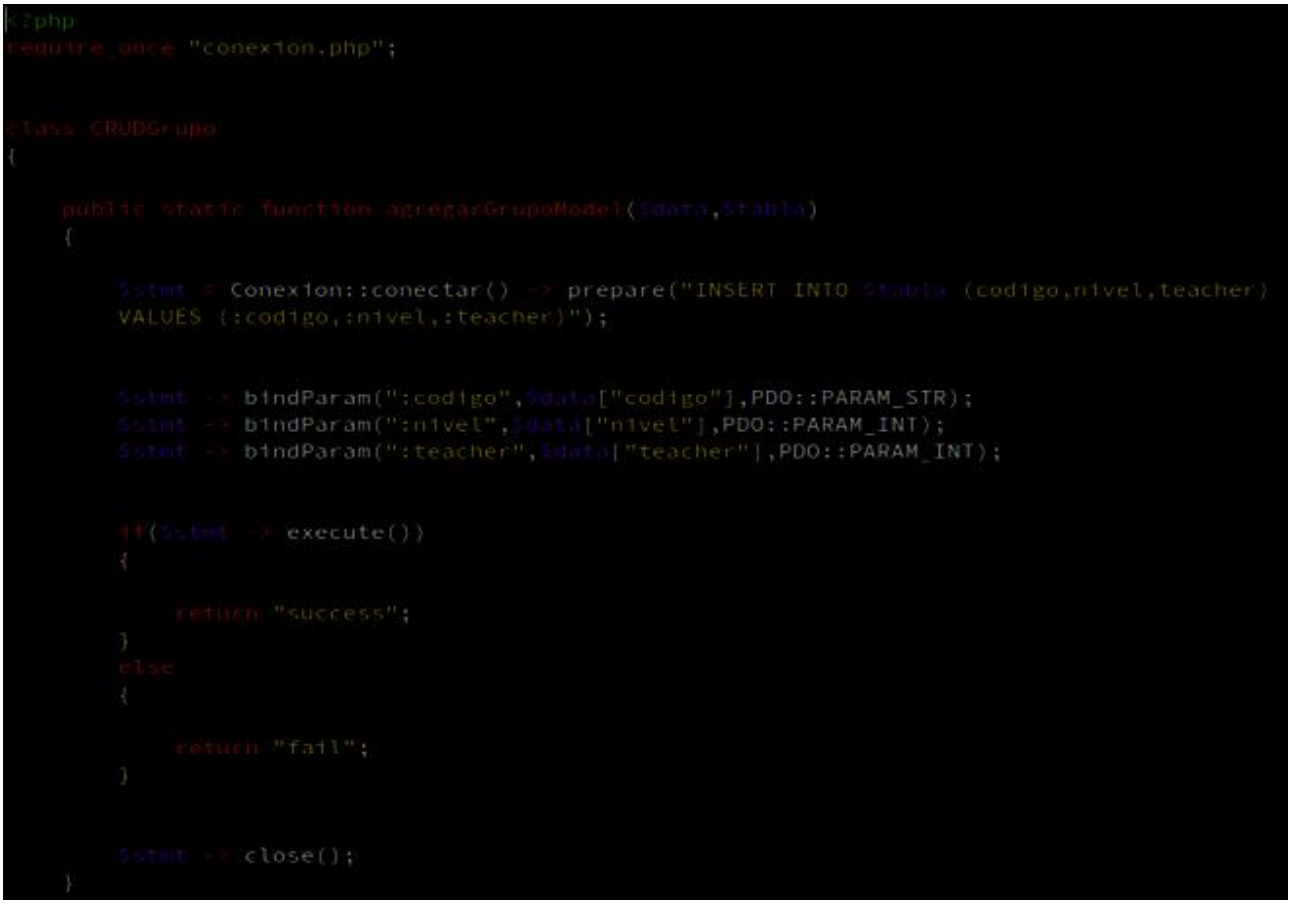


```
    public function optionGrupoTeacherController()
    {
        $data = CRUDGrupo::listadoGrupoModel("grupo","teacher","usuario");

        foreach($data as $rows => $row)
        {
            echo "<option class='repairtext' value=".$row["codigo"].">".$row["teacher"]."
            - ".$row["codigo"]."</option>";
        }
    }
}
?>
```

Este controlador recibe la información de los teacher y los grupos que le pertenece a cada teacher para después mostrar su información mediante un select.

CRUD DE GRUPOS (crudGrupo.php)

MODELO PARA AGREGAR UN GRUPO



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


class CRUDGrupo
{

    public static function agregarGrupoModel($data,$tabla)
    {

        $stmt = Conexion::conectar()->prepare("INSERT INTO $tabla (codigo,nivel,teacher)
        VALUES (:codigo,:nivel,:teacher)");


        $stmt->bindParam(":codigo",$data["codigo"],PDO::PARAM_STR);
        $stmt->bindParam(":nivel",$data["nivel"],PDO::PARAM_INT);
        $stmt->bindParam(":teacher",$data["teacher"],PDO::PARAM_INT);


        if($stmt->execute())
        {

            return "success";
        }
        else
        {

            return "fail";
        }


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

Este modelo recibe del controlador, la información de un nuevo grupo, el cual mediante este modelo se registrara en la base de datos.

MODELO PARA LISTAR LOS GRUPOS

```
public static function listadoGrupoModel($tabla1,$tabla2,$tabla3)
{
    $stmt = Conexion::conectar() -> prepare("SELECT g.codigo as codigo , g.nivel as
    nivel , u.nombre as teacher
                                            FROM $tabla1 as g
                                            LEFT JOIN $tabla2 as t on t.teacher =
                                            g.teacher
                                            LEFT JOIN $tabla3 as u on u.num_empleado
                                            = t.teacher");

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Este modelo obtiene toda la información de los grupos registrados en la base de datos, después de obtener la información de los grupos se le retorna esta información al controlador para ser mostrada.

MODELO PARA EDITAR EL GRUPO

```
public static function editarGrupoModel($data,$tabla1,$tabla2,$tabla3)
{
    $stmt = Conexion::conectar()->prepare("SELECT g.codigo as codigo, g.nivel as
    nivel, u.nombre as teacher, u.num_empleado as id
                                        FROM $tabla1 as g
                                        LEFT JOIN $tabla2 as t on t.teacher =
                                        g.teacher
                                        LEFT JOIN $tabla3 as u on u.num_empleado =
                                        t.teacher
                                        WHERE g.codigo = :id");

    $stmt->bindParam(":id",$data, PDO::PARAM_STR);

    $stmt->execute();

    return $stmt->fetch();

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

Este modelo obtiene la información de un grupo en especifico registrado en la base de datos para ser retornada al controlador para poder ser mostrada y editada.

MODELO PARA LISTAR LOS GRUPOS EN UN SELECT

```php
public static function optionGrupoModel($tabla)
{

    $stmt = Conexion::conectar() -> prepare("SELECT codigo FROM $tabla");


    $stmt -> execute();


    return $stmt -> fetchAll();


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

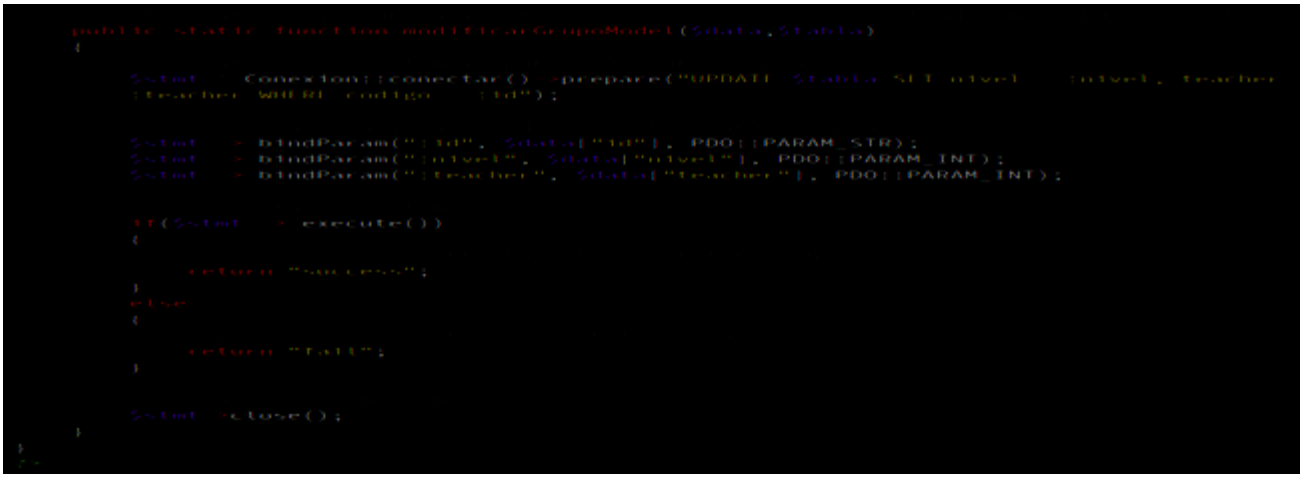
Este modelo obtiene el nombre de todos los grupos registrados en la base de datos del sistema, para después retornarlos al controlador para ser mostrados en un select.

MODELO PARA ELIMINAR UN GRUPO

```php
public static function eliminarGrupoModel($data,$tabla1,$tabla2)
{

    $stmt1 = Conexion::conectar() -> prepare("UPDATE $tabla1 SET grupo = NULL WHERE
    grupo = :id");


    $stmt1 -> bindParam(":id",$data,PDO::PARAM_STR);


    $stmt2 = Conexion::conectar() -> prepare("DELETE FROM $tabla2 WHERE codigo =
    :id");


    $stmt2 -> bindParam(":id",$data,PDO::PARAM_STR);


    if($stmt1 -> execute() && $stmt2 -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


    $stmt1 -> close();
    $stmt2 -> close();
}
```

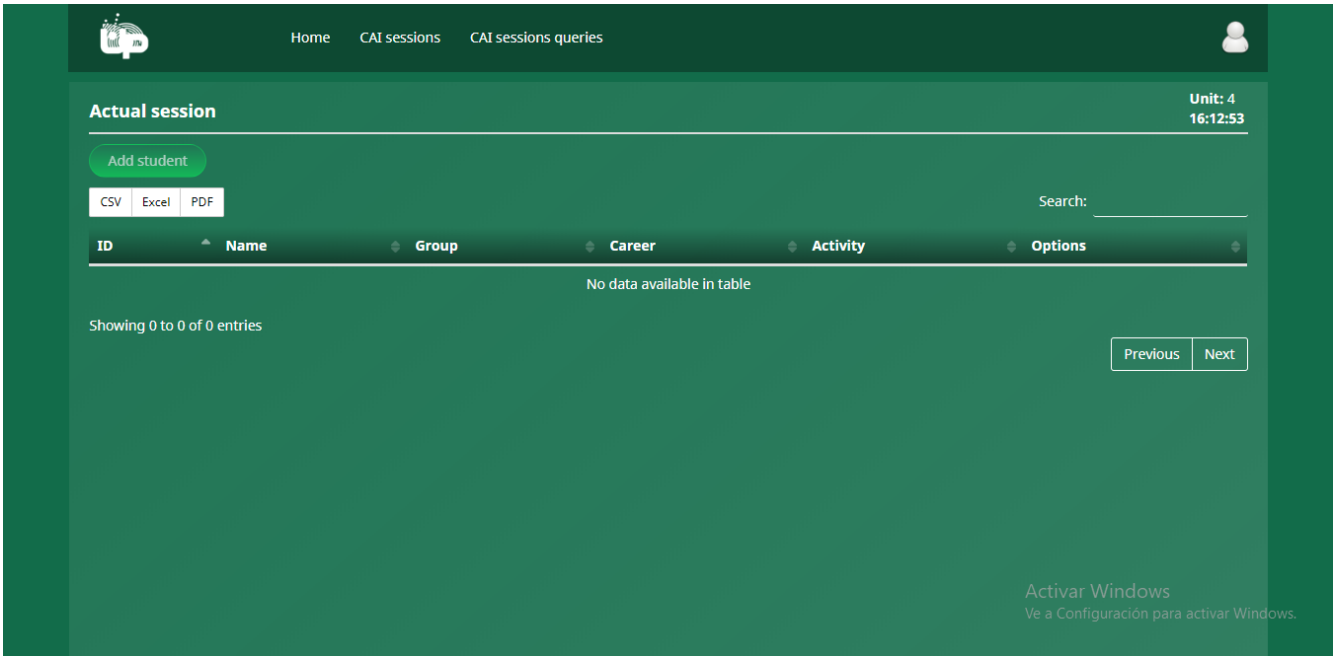
Este modelo recibe del controlador el id de un grupo a ser eliminado de la base de datos, una vez recibido el id del grupo el modelo elimina la información del grupo y la información que este relacionada con el grupo a eliminar.

MODELO PARA MODIFICAR UN GRUPO



Este modelo recibe la información modificada de un grupo, una vez recibida esta información, actualiza la información existente del grupo con la información recibida.

MÓDULO DE SESIÓN



CONTROLADOR DE SESIONES (controllerSession.php)

CONTROLADOR PARA MOSTRAR LAS SESIONES

```
public function mostrarSessionController($id, $actividad)
{

    $data = $id;


    $resp = CRUDSession::mostrarSessionModel($data,"alumno","carrera");


    echo "
          <input type=hidden value=".$actividad." name='actividad'>
          <img class='pull-left' width='400px' height='400px' src='".$resp[0]["img"]."'/>
          <div class='text-black' style='margin-left: 420px'>
            <p>
                <b>ID:</b>
                <br>
                <input class='form-control' type='text' name='matricula' value='".$resp[0]["matricula"]."'
                    readonly>
            </p>
            <p>
              <b>Name:</b>
              <br>
              <input class='form-control' type='text' name='nombre' value='".$resp[0]["nombre"]." ".$resp[0
                  ]["apellido"]."' readonly>
            </p>
            <p>
              <b>Group:</b>
              <br>
              <input class='form-control' type='text' name='grupo' value='".$resp[0]["grupo"]."' readonly>
            </p>
            <p>
              <b>Career:</b>
```

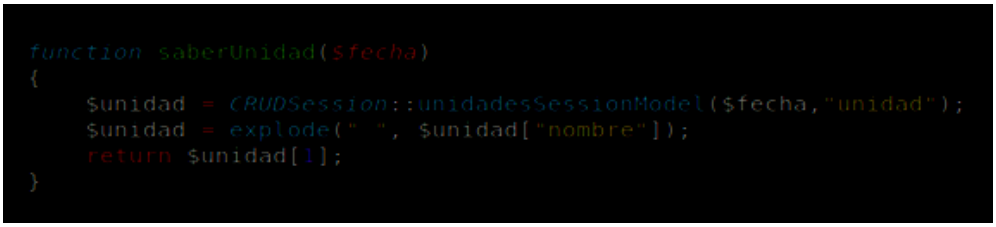
Se muestra la información del alumno que desea ingresar a la sesión que se encuentra activa, con sesión se refiere a la hora en que puede hacer CAI.

CONTROLADOR PARA AGREGAR UNA SESIÓN

```
function agregarSessionController()
{
    if(isset($_POST["matricula"]))
    {
        $fecha = date("Y-m-d");
        $horaE = date("H:i:s");
        $unidad = CRUDSession::unidadesSessionModel($fecha,"unidad");
        $grupo = CRUDSession::grupoSessionModel($_POST["grupo"],"grupo","usuario");
        $data = array("matricula" => $_POST["matricula"],
                      "fecha" => $fecha,
                      "horaE" => $horaE,
                      "actividad" => $_POST["actividad"],
                      "unidad" => $unidad["id_unidad"],
                      "nivel" => $grupo["nivel"],
                      "teacher" => $grupo["teacher"],
                      "grupo" => $_POST["grupo"]);

        $resp = CRUDSession::agregarSessionModel($data,"asistencia");

        if($resp == "success")
        {
            $_SESSION["mensaje"] = "add";

            echo "<script>
                    window.location.replace('index.php?section=sessions&action=actual');
```

Para agregar un alumno a la sesión se toman sus datos y se almacenan un arreglo junto con la hora del sistema y la fecha, se envían al modelo y este es el encargado de registrarlo, se redireccionará a la vista donde se listan los alumnos en esa sesión si todo fue realizado correctamente y mostrará un mensaje diciendo que el alumno se agregó correctamente.

CONTROLADOR QUE OBTIENE LA UNIDAD



```
function saberUnidad($fecha)
{
    $unidad = CRUDSession::unidadesSessionModel($fecha,"unidad");
    $unidad = explode(" ", $unidad["nombre"]);
    return $unidad[1];
}
```

Se envía la fecha al modelo para saber en qué unidad se encuentra actualmente.

CONTROLADOR PARA LISTAR LOS ALUMNOS EN LA SESIÓN

```
function listadoSessionController()
{
    $data = CRUDSession::listadoSessionModel("asistencia","alumno","grupo","carrera","actividad");

    foreach($data as $rows => $row)
    {
        echo "<tr class='fondoTabla'>
            <td>".$row["asistencia"]."</td>
            <td>".$row["nombre"]." ".$row["apellido"]."</td>
            <td>".$row["grupo"]."</td>
            <td>".$row["carrera"]."</td>
            <td>".$row["actividad"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' id='eliminar' data-toggle='modal'
                        data-target='#delete-modal' onclick='idDel(".$row["asistencia"].",".$row["
                        hora_entrada"].")'>Finish</button>
                </center>
            </td>
        </tr>";
    }
}
```

Es por medio de una tabla que se muestran los alumnos que se encuentran en esa sesión actualmente.

CONTROLADOR PARA MOSTRAR EL HISTORIAL DE LA SESIÓN

```
public function historialSessionController()
{
    $data = array("grupo" => "",
                  "unidad" => "");

    if(isset($_POST))
    {
        if(!empty($_POST["grupo"]))
        {
            $data["grupo"] = $_POST["grupo"];
        }

        if(!empty($_POST["unidad"]))
        {
            $data["unidad"] = $_POST["unidad"];
        }
    }

    $resp =
    CRUDSession::historialSessionModel($data,"usuario","teacher","grupo","alumno");

    foreach($resp as $rows => $row)
    {
        $hours =
        CRUDSession::horasModel($row["matricula"],$_POST["unidad"],"asistencia","unida
        d");

        echo
        "
        <tr class='fondoTabla'>
            <td>".$row["matricula"]."</td>
            <td>".$row["nombre"]." ".$row["apellido"]."</td>
            <td>Level ".$row["nivel"]."</td>
```

Este controlador se utiliza para conocer cuantas horas de CAI han realizados los alumno de un grupo. Para esto recibe del sistema el grupo y unidad que le enviara al modelo para obtener dicha información, una vez obtenida la información, la muestra en una tabla.

CONTROLADOR DE LAS HORAS DE CAI

```
    function horasCAIController()
    {
        $data =
        CRUDSession::horasCAIModel($_GET["student"],$_GET["group"],$_GET["unit"],"asistenc
        ia","unidad","actividad","alumno");

        foreach($data as $rows => $row)
        {
            echo
            "
            <tr class='fondoTabla'>
                <td>".$row["nombre"]." ".$row["apellido"]."</td>
                <td>".$row["fecha"]."</td>
                <td>".$row["hora_entrada"]."</td>
                <td>".$row["hora_salida"]."</td>
                <td>".$row["actividad"]."</td>
                <td>".$row["unidad"]."</td>
            </tr>
            ";
        }
    }
}
?>
```

Este controlador recibe del modela la información detallada de las horas de CAI realizadas por un alumno en una determinada unidad. Una vez obtenida esta información, la muestra en una tabla.

CRUD DE SESIONES (crudSession.php)

MODELO PARA MOSTRAR LA INFORMACIÓN DEL ALUMNO

```
public static function mostrarSessionModel($data,$tabla, $tabla2)
{

    $stmt = Conexion::conectar() -> prepare("SELECT a.img, a.matricula as matricula, a.nombre as nombre,
                                        a.apellido as apellido, a.grupo as grupo,
                                        c.nombre as carrera
                                            FROM $tabla as a
                                            JOIN $tabla2 as c on c.siglas = a.carrera
                                            WHERE matricula = :id ");


    $stmt -> bindParam(":id",$data,PDO::PARAM_INT);

    $stmt -> execute();

    return $stmt -> fetchAll();

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

Retorna la información del alumno.

MODELO PARA SABER LA UNIDAD

```
public static function unidadesSessionModel($data,$tabla)
{

    $stmt = Conexion::conectar()->prepare("SELECT * FROM $tabla WHERE :fecha >= fecha inicio AND :fecha <=
        fecha fin");


    $stmt->bindParam(":fecha",$data, PDO::PARAM_STR);


    $stmt->execute();


    return $stmt->fetch();


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

Se compara la fecha dada entre los rangos que se encuentran registrados y se retorna la información de la unidad.

MODELO PARA OBTENER EL NIVEL Y TEACHER DEL ALUMNO

```
public static function grupoSessionModel($data,$tabla1,$tabla2)
{

    $stmt = Conexion::conectar()->prepare("SELECT g.nivel as nivel, u.nombre as teacher
                                           FROM $tabla1 as g
                                           JOIN $tabla2 as u on u.num_empleado = g.teacher
                                           WHERE codigo = :id");


    $stmt->bindParam(":id",$data, PDO::PARAM_STR);


    $stmt->execute();


    return $stmt->fetch();


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

De acuerdo al id del alumno se busca su nivel de inglés, el teacher que da ese nivel de inglés y el código del grupo.

MODELO PARA AGREGAR UNA SESIÓN

```php
public static function agregarSessionModel($data,$tabla)
{

    $stmt = Conexion::conectar() -> prepare("INSERT INTO $tabla(
        id asistencia,fecha,hora entrada,alumno,actividad,unidad,nivel,teacher,grupo)
        VALUES (NULL,:fecha,:horaE,:alumno,:actividad,:unidad,:nivel,:teacher,:grupo)");


    $stmt -> bindParam(":fecha",$data["fecha"],PDO::PARAM_STR);
    $stmt -> bindParam(":horaE",$data["horaE"],PDO::PARAM_STR);
    $stmt -> bindParam(":alumno",$data["matricula"],PDO::PARAM_INT);
    $stmt -> bindParam(":actividad",$data["actividad"],PDO::PARAM_INT);
    $stmt -> bindParam(":unidad",$data["unidad"],PDO::PARAM_INT);
    $stmt -> bindParam(":nivel",$data["nivel"],PDO::PARAM_INT);
    $stmt -> bindParam(":teacher",$data["teacher"],PDO::PARAM_STR);
    $stmt -> bindParam(":grupo",$data["grupo"],PDO::PARAM_STR);


    if($stmt -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }
```

Se obtienen los datos para que el registro sea agregado a la base de datos. Retornando “success” si la inserción fue correcta y “fail” en caso contrario.

MODELO PARA LISTAR LOS ALUMNOS EN LA SESIÓN

```php
public static function listadoSessionModel($tabla1,$tabla2, $tabla3, $tabla4, $tabla5)
{

    $stmt = Conexion::conectar() -> prepare("SELECT asi.id asistencia as asistencia, a.nombre as nombre, a.
        apellido as apellido, g.codigo as grupo, c.nombre as carrera, act.nombre as actividad, asi.
        hora entrada
                                                FROM $tabla1 as asi
                                                JOIN $tabla2 as a on a.matricula = asi.alumno
                                                JOIN $tabla3 as g on g.codigo = asi.grupo
                                                JOIN $tabla4 as c on c.siglas = a.carrera
                                                JOIN $tabla5 as act on act.id actividad = asi.actividad
                                                    WHERE hora salida IS NULL");


    $stmt -> execute();


    return $stmt -> fetchAll();


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

Se obtienen los datos de los alumnos para mostrarlo en una tabla.

MODELO PARA EL HISTORIAL DE LA SESIÓN

```
public static function historialSessionModel($data,$tabla1,$tabla2,$tabla3,$tabla4)
{

    $stmt = Conexion::conectar() -> prepare("SELECT a.matricula, a.nombre, a.apellido,
    u.nombre as teacher, g.codigo as grupo, g.nivel
                                            FROM $tabla1 as u
                                            JOIN $tabla2 as t on t.teacher =
                                            u.num_empleado
                                            JOIN $tabla3 as g on g.teacher =
                                            t.teacher
                                            JOIN $tabla4 as a on a.grupo = g.codigo
                                            WHERE g.codigo = '".$data["grupo"]."'");


    $stmt -> execute();


    return $stmt -> fetchAll();


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

Este modelo obtiene de la base de datos la información de los alumnos que pertenecen a un determinado grupo. Una vez que obtiene esta información la retorna al controlador para ser mostrada.

MODELO PARA LAS HORAS DE CAI

```
public static function horasModel($student,$unit,$tabla1,$tabla2)
{

    $stmt = Conexion::conectar() -> prepare("SELECT COUNT(*) as horas
                                            FROM $tabla1 as a
                                            JOIN $tabla2 as u on u.id_unidad =
                                            a.unidad
                                            WHERE a.alumno = :matricula && unidad =
                                            :unidad && hora_completa = 1");


    $stmt -> bindParam(":matricula",$student,PDO::PARAM_INT);
    $stmt -> bindParam(":unidad",$unit,PDO::PARAM_INT);


    $stmt -> execute();


    return $stmt -> fetch();


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

Este modelo obtienen el total de horas de CAI realizadas por un alumno en una determinada unidad. Una vez que obtiene esta información se la retorna al controlador pare ser mostrada.

MODELO PARA LAS HORAS DE CAI DEL ALUMNO

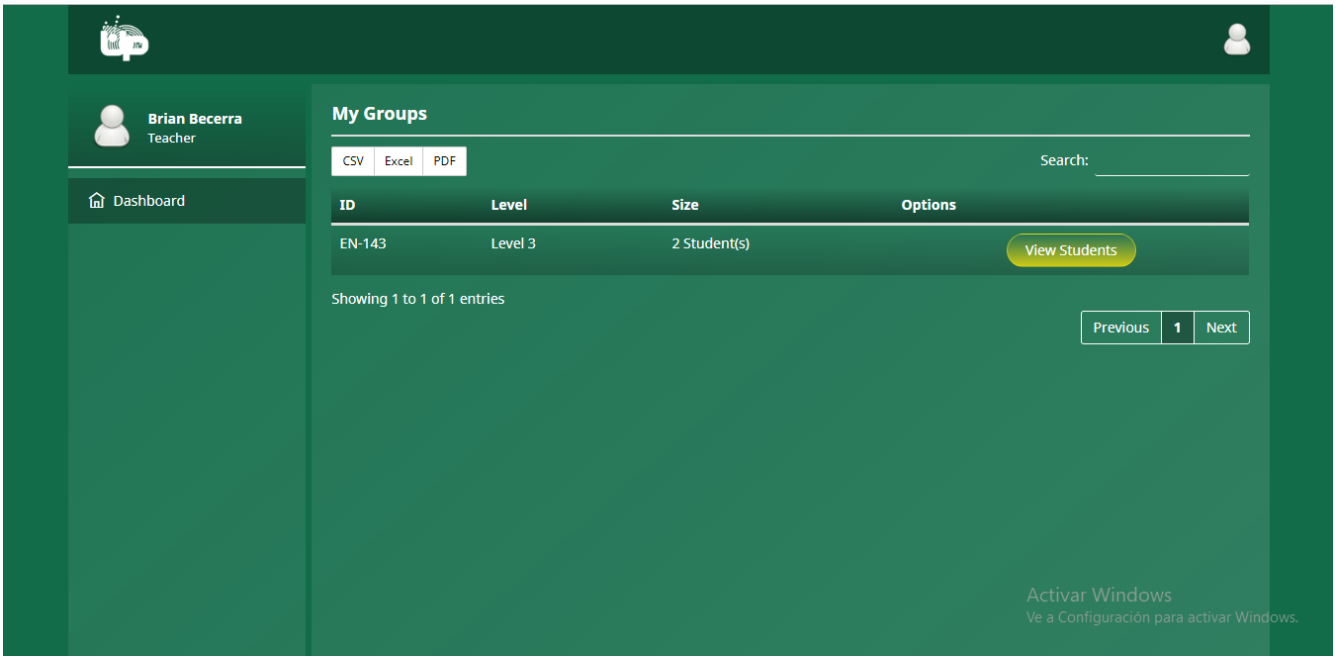
```
public static function
horasCAIModel($student,$group,$unit,$stable1,$stable2,$stable3,$stable4)
{

    $stmt = Conexion::conectar() -> prepare("SELECT al.nombre, al.apellido, a.fecha,
    a.hora_entrada, a.hora_salida, u.nombre as unidad, ac.nombre as actividad
                                            FROM stable1 as a
                                            JOIN stable2 as u on u.id_unidad =
                                            a.unidad
                                            JOIN stable3 as ac on ac.id_actividad =
                                            a.actividad
                                            JOIN stable4 as al on al.matricula =
                                            a.alumno
                                            WHERE a.alumno = :matricula &&
                                            u.id_unidad = :unidad && a.hora_completa
                                            = 1 && a.grupo = :grupo");


    $stmt -> bindParam(":matricula",$student,PDO::PARAM_INT);
    $stmt -> bindParam(":grupo",$group,PDO::PARAM_STR);
    $stmt -> bindParam(":unidad",$unit,PDO::PARAM_INT);


    $stmt -> execute();


    return $stmt -> fetchAll();


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

Este controlador obtiene la información detallada, de la base de datos, de las horas de CAI realizadas por un determinado alumno. Una vez que obtiene esta información la retorna la modelo para ser mostrada.

```
    public static function terminar()
    {
        $hora = Date("H:i:s");
        $completa = 1;

        $stmt = Conexion::conectar()->prepare("UPDATE asistencia SET hora_salida = :horaS,
        hora_completa = :completa WHERE hora_salida IS NULL");


        $stmt -> bindParam(":horaS", $hora, PDO::PARAM_STR);
        $stmt -> bindParam(":completa",$completa, PDO::PARAM_INT);


        if($stmt -> execute())
        {

            return "success";
        }
        else
        {

            return "fail";
        }


        $stmt->close();

    }
}
?>
```

El modelo **terminar:** este modelo es llamado automáticamente cada determinado tiempo, se utiliza para terminar de forma automática todas las horas de CAI que se están realizando en las sesión actual.

MÓDULO DE TEACHERS



CONTROLADOR DE TEACHERS(controllerTeacher.php)

Este archivo contiene la clase de **mvcTeacher**, esta clase contiene los controladores los cuales maneja todo lo relacionado con la sección de Teacher.

```php
<?php

class mvcTeacher
{

    function listadoGrupoTeacherController()
    {

        $data = CRUDTeacher::listadoGrupoTeacherModel($_SESSION["empleado"],"grupo");


        foreach($data as $rows => $row)
        {
            $size =
            CRUDAlumno::listadoAlumnoGrupoModel($row["codigo"],"alumno","carrera");


            echo "<tr class='fondoTabla'>
                <td>".$row["codigo"]."</td>
                <td>Level ".$row["nivel"]."</td>
                <td>".count($size)." Student(s)</td>
                <td>
                    <center>
                        <a href='index.php?section=groups&action=my-
                        students&group=".$row["codigo"]."'><button class='btn btn-rounded
                        btn-warning' type='button' name='button'>
                            View Students
                        </button></a>
                    </center>
                </td>
            </tr>";
        }
    }
```

El primer controlador es **listadoGrupoTeacherController:** este controlador recibe del modelo la información de los grupos que pertenecen a un teacher, para después mostrar la información en una tabla.

```
function idGrupoAlumno()
{

    $data = CRUDTeacher::dataAlumnoModel($_GET["student"],"alumno");

    return $data["grupo"];
}
```

El controlador **idGrupoAlumno:** este controlador recibe mediante el modelo el nombre del grupo al que pertenece un alumno.

```
function dataAlumnoController()
{

    $data = CRUDTeacher::dataAlumnoModel($_GET["student"],"alumno");

    $name = CRUDTeacher::nombreCarreraModel($data["carrera"],"carrera");

    echo
        "
    <img class='pull left' width='400px' height='400px' src='".$data["img"]."'>
    <div class='text-white' style='margin-left: 420px'>
        <p>
            <b>ID:</b><br>".$data["matricula"]."
        </p>
        <p>
            <b>Name:</b><br>".$data["nombre"]." ".$data["apellido"]."
        </p>
        <p>
            <b>Group:</b><br>".$data["grupo"]."
        </p>
        <p>
            <b>Career:</b><br>".$name["nombre"]."
        </p>
    </div>
    ";
}
```

El controlador **dataAlumnoController:** este controlador recibe la información de un alumno para después ser mostrada en el sistema.

```
    function horasAlumnoController()
    {

        $data =
        CRUDTeacher::horasAlumnoModel($_GET["student"],$_GET["group"],"asistencia","unidad
        ","actividad");

        foreach($data as $rows => $row)
        {

            echo
            "
            <tr class='fondoTabla'>
                <td>".$row["fecha"]."</td>
                <td>".$row["hora_entrada"]."</td>
                <td>".$row["hora_salida"]."</td>
                <td>".$row["actividad"]."</td>
                <td>".$row["unidad"]."</td>
            </tr>
            ";
        }
    }
}
?>
```

El controlador **horasAlumnoController:** este controlador mediante el modelo recibe la información de las horas de CAI que el alumno ha realizados, na vez recibida la información de la horas, las muestra en una tabla.

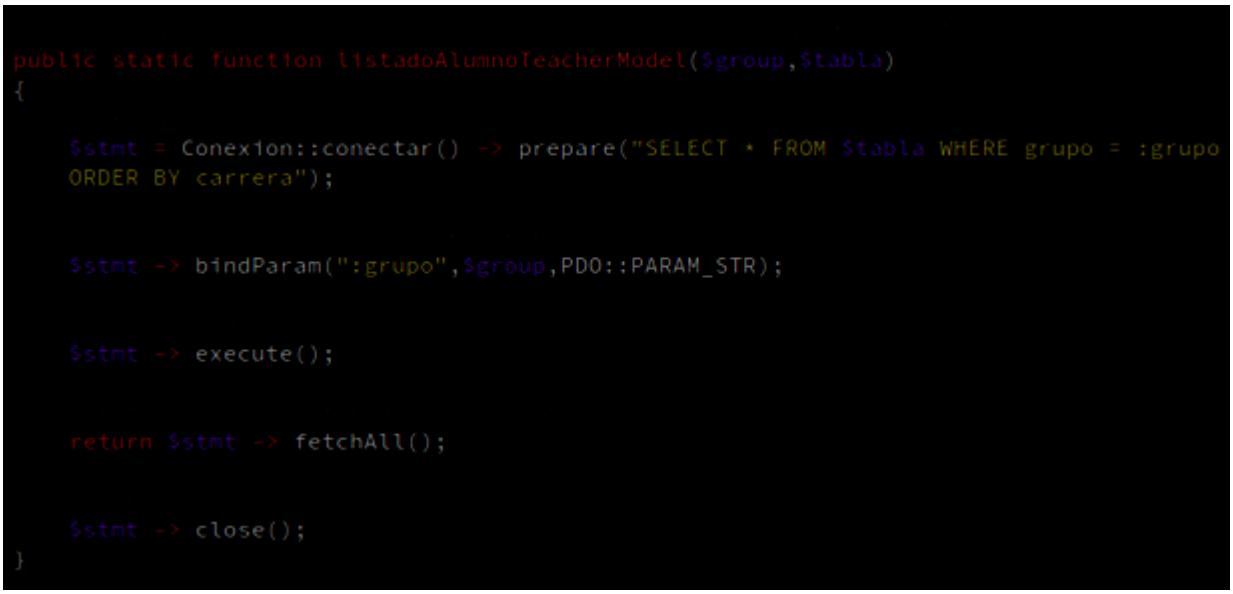
Por ultimo esta el controlador **listadoAlumnoTeacherController:** este controlador recibe la información de los alumnos que pertenecen a un grupo de un teacher, para después mostrar la información de los alumnos en tablas, separados por carrera.

CRUD DE TEACHERS(crudTeacher.php)

Este archivo contiene la clase de **CRUDTeacher**, esta clase contiene los modelos para manejar las operaciones en la base de datos para la sección de teacher del sistema.

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


class CRUDTeacher
{

    public static function listadoGrupoTeacherModel($teacher,$tabla)
    {

        $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla WHERE teacher =
        :teacher");


        $stmt -> bindParam(":teacher",$teacher,PDO::PARAM_INT);


        $stmt -> execute();


        return $stmt -> fetchAll();


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

El primer controlador es **listadoGrupoTeacher:** este modelo obtiene de la base de datos la información de los grupos de un teacher, para después ser retornada al controlador y la información pueda ser mostrada en el sistema.



```php
public static function listadoAlumnoTeacherModel($group,$tabla)
{

    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla WHERE grupo = :grupo
    ORDER BY carrera");


    $stmt -> bindParam(":grupo",$group,PDO::PARAM_STR);


    $stmt -> execute();


    return $stmt -> fetchAll();


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

El modelo **listadoAlumnoTeacherModel:** este modelo obtienen la información de la base de datos, de todos los alumnos que pertenecen a un determinado grupo, para después retornar la información y pueda ser mostrada

```php
public static function dataAlumnoModel($student,$tabla)
{
    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla WHERE matricula =
    :matricula");

    $stmt -> bindParam(":matricula",$student,PDO::PARAM_INT);

    $stmt -> execute();

    return $stmt -> fetch();

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

El modelo **dataAlumnoModel:** este modelo obtiene la información de un alumno de la base de datos, para después ser retornada al controlador y ser mostrada en el sistema.

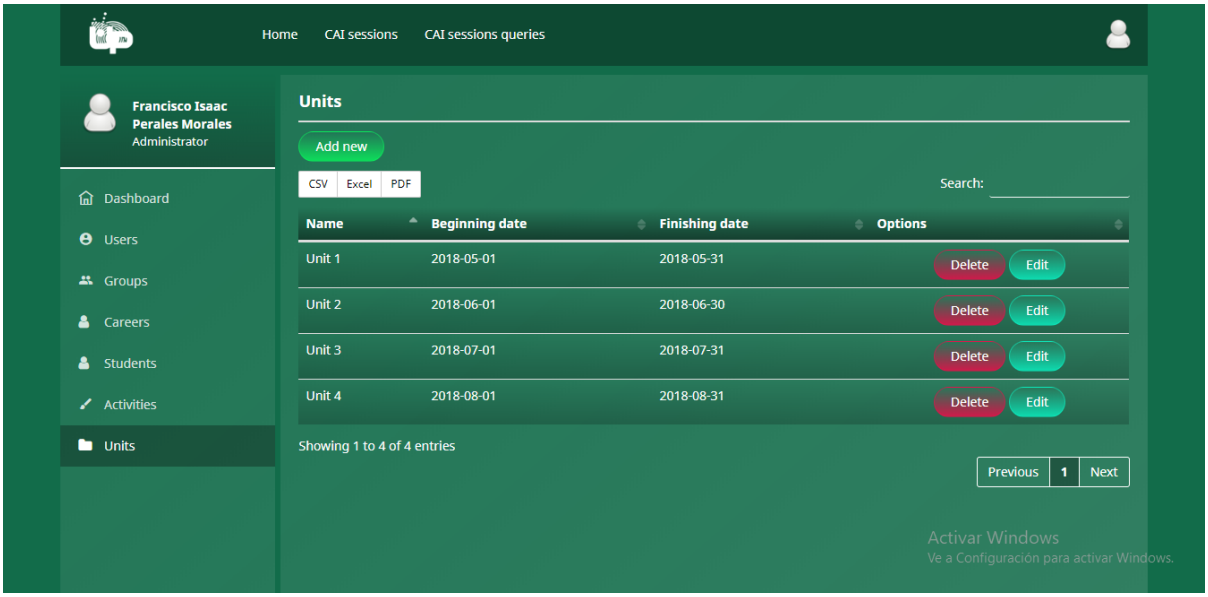
```php
public static function horasAlumnoModel($student,$group,$tabla1,$tabla2,$tabla3)
{
    $stmt = Conexion::conectar() -> prepare("SELECT a.fecha, a.hora_entrada,
    a.hora_salida, u.nombre as unidad, ac.nombre as actividad
                                            FROM $tabla1 as a
                                            JOIN $tabla2 as u on u.id_unidad =
                                            a.unidad
                                            JOIN $tabla3 as ac on ac.id_actividad =
                                            a.actividad
                                            WHERE a.alumno = :matricula &&
                                            a.hora_completa = 1 && grupo = :grupo");

    $stmt -> bindParam(":matricula",$student,PDO::PARAM_INT);
    $stmt -> bindParam(":grupo",$group,PDO::PARAM_STR);

    $stmt -> execute();

    return $stmt -> fetchAll();

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

El modelo **horasAlumnoModel:** ese modelo obtiene la información de la base de datos de todas las horas de CAI que haya realizado un determinado alumno, para después retorna la información de las hora y sea mostrada por el controlador en el sistema.

```php
    //modelo para obtener el nombre de la carrera
    public static function nombreCarreraModel($carrera,$tabla)
    {
        //preparar la consulta
        $stmt = Conexion::conectar() -> prepare("SELECT nombre FROM $tabla WHERE siglas =
        :siglas");

        //asignamos los datos para el select
        $stmt -> bindParam(":siglas",$carrera,PDO::PARAM_STR);

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

        //retornamos la información de la tabla
        return $stmt -> fetch();

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

Por ultimo esta el modelo **nombreCarreraModel:** este modelo recibe el id de una carrera, una bes que haya recibido el id de la carrera, de la base de datos obtiene en nombre de la carrera, para después retornarla a controlador para ser mostrada.

MÓDULO DE UNIDADES



CONTROLADOR DE UNIDADES(controllerUnidad.php)

Este archivo contiene la clase **mvcUnidad** que tiene los controladores los cuales maneja todo lo relacionado con la sección de Unidad.

Primero tenemos al controlador **agregarUnidadController:** este controlador se encarga de recibir la información de una nueva unidad a registrar en el sistema. Una vez recibida la información la almacena en un array y la enviá al modelo para que sea registrada en la base datos.

```php
function listadoUnidadController()
{

    $data = CRUDUnidad::listadoUnidadModel("unidad");

    foreach($data as $rows => $row)
    {

        echo "<tr class='fondoTabla'>
            <td>".$row["nombre"]."</td>
            <td>".$row["fecha_inicio"]."</td>
            <td>".$row["fecha_fin"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' data-toggle='modal'
                    data-target='#delete-modal'
                    onclick='idDel(".$row["id_unidad"].")'> Delete </button>

                    <a href='index.php?
                    section=units&action=list&edit=".$row["id_unidad"]."'>
                        <button class='btn btn-rounded btn-custom'>Edit</button>
                    </a>
                </center>
            </td>
            </tr>";
    }
}
```

El controlador **listadoUnidadController:** este controlador recibe del modelo la información de la unidades registradas en la base de datos, una vez recibida esta información la muestra mediante una tabla.

```php
public function eliminarUnidadController()
{

    if(isset($_POST["del"]))
    {

        $data = $_POST["del"];

        $resp = CRUDUnidad::eliminarUnidadModel($data,"asistencia","unidad");

        if($resp == "success")
        {

            $_SESSION["mensaje"] = "delete";

            echo "<script>
                    window.location.replace('index.php?section=units&action=list');
                  </script>";
        }
    }
}
```

El controlador **eliminarUnidadController:** este controlador recibe del sistema el id de la unidad a eliminar de la base de datos. Una vez recibido el id se lo enviá al modelo para su borrado en la base de datos.

```php
//Control para poder mostrar la información de una unidad a editar
public function editarUnidadController()
{
    //se obtiene el id del unidad a mostrar su información
    $data = $_GET["edit"];

    //se envía el id de la unidad y el nombre de la tabla donde esta ubicada
    $resp = CRUDUnidad::editarUnidadModel($data,"unidad");

    //se muestra la información de la unidad en inputs de un formulario
    echo
        "
            <input type='hidden' name='id' value=".$resp["id_unidad"].">

            <div class='row'>
                <div class='col-md-12'>
                    <div class='form-group'>
                        <label class='control-label repairtext'>Name</label>
                        <input type='text' name='nombre' class='form-control'
                        value='".$resp["nombre"]."' placeholder='Name'>
                    </div>
                </div>
            </div>
            <div class='row'>
                <div class='col-md-12'>
                    <div class='form-group'>
                        <label class='control-label repairtext'>Beginning date</label>
                        <input type='date' name='inicio' class='form-control'
                        value=".$resp["fecha_inicio"].">
                    </div>
                </div>
            </div>
            <div class='row'>
                <div class='col-md-12'>
                    <div class='form-group'>
                        <label class='control-label repairtext'>Finishing date</label>
                        <input type='date' name='fin' class='form-control'
                        value=".$resp["fecha_fin"].">
                    </div>
                </div>
            </div>
    ";
}
```

El controlador **editarUnidadController:** este controlador recibe del modelo la información de una unidad en especifica, una vez recibida, la muestra en un formulario para que pueda ser editada en caso de ser necesario.

```php
public function modificarUnidadController()
{
    if(isset($_POST["id"]))
    {
        $data = array("id" => $_POST["id"],
                      "nombre" => $_POST["nombre"],
                      "inicio" => $_POST["inicio"],
                      "fin" => $_POST["fin"]);

        $resp = CRUDUnidad::modificarUnidadModel($data,"unidad");

        if($resp == "success")
        {
            $_SESSION["mensaje"] = "edit";

            echo "<script>
                    window.location.replace('index.php?section=units&action=list');
                  </script>";
        }
    }
}
```

El controlador modificarUnidadController: este controlador recibe la información modificada de una unidad, después esta información se la enviá al modelo para actualizar la información de la unidad en la base de datos.

```php
    public function optionUnidadController()
    {
        $data = CRUDUnidad::optionUnidadModel("unidad");

        foreach($data as $rows => $row)
        {
            echo "<option class='repairtext'
            value=".$row["id_unidad"].">".$row["nombre"]."</option>";
        }
    }
}
?>
```

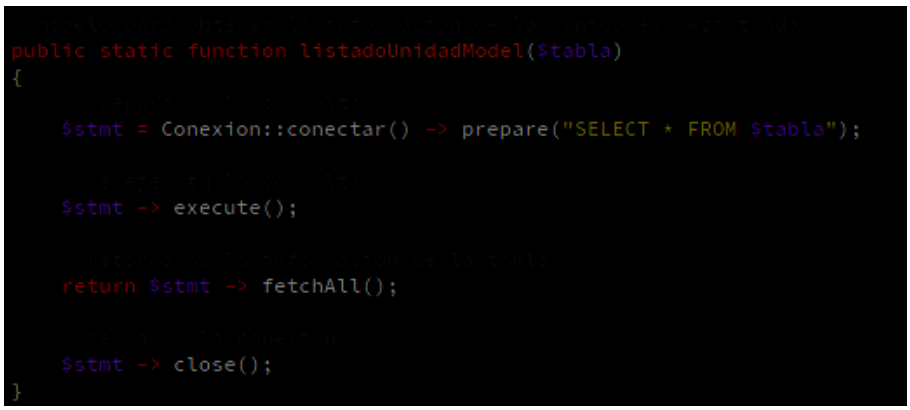
Por ultimo está el controlador **optionUnidadController:** este controlador recibe la información de todas las unidades, para después mostrar la información de las unidades mediante un select.

CRUD DE UNIDADES(crudUnidad.php)

Este archivo contiene la clase de **CRUDUnidad**, esta clase contiene los modelos para manejar las operaciones en la base de datos para la sección de unidad del sistema.

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


class CRUDUnidad
{

    public static function agregarUnidadModel($data,$tabla)
    {

        $stmt = Conexion::conectar() -> prepare("INSERT INTO $tabla
        (nombre,fecha_inicio,fecha_fin) VALUES (:nombre,:inicio,:fin)");


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


        if($stmt -> execute())
        {

            return "success";
        }
        else
        {

            return "fail";
        }


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

Primero tenemos al modelo **agregarUnidadModel:** este modelo recibe del controlador, la información de una nueva unidad, el cual mediante este modelo se registrara en la base de datos.



```php
public static function listadoUnidadModel($tabla)
{

    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla");


    $stmt -> execute();


    return $stmt -> fetchAll();


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

El modelo **listadoUnidadModel:** este modelo obtinene la información de todas las unidades registradas en la base de datos. Una vez obtenida la información, la retorna al controlador para que la información pueda ser mostrada.

```
public static function eliminarUnidadModel($dato,$tabla1,$tabla2)
{

    $stmt1 = Conexion::conectar() -> prepare("DELETE FROM $tabla1 WHERE unidad =
    :id");


    $stmt1 -> bindParam(":id",$dato,PDO::PARAM_INT);


    $stmt2 = Conexion::conectar() -> prepare("DELETE FROM $tabla2 WHERE id_unidad =
    :id");


    $stmt2 -> bindParam(":id",$dato,PDO::PARAM_INT);


    if($stmt1 -> execute() && $stmt2 -> execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


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

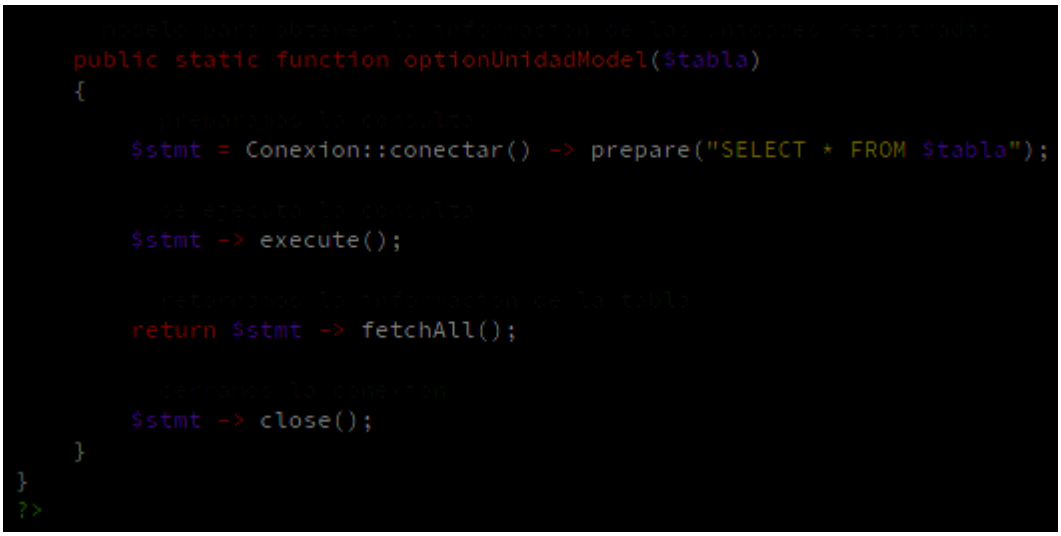
El modelo **eliminarUnidadModel:** este modelo recibe del controlador el id de una unidad que debe ser eliminada. Una vez recibido el id, el modelo borra la unidad de la base de datos y toda la información relacionada con esta unidad.

```
public static function editarUnidadModel($data,$tabla)
{

    $stmt = Conexion::conectar()->prepare("SELECT * FROM $tabla WHERE id_unidad =
    :id");


    $stmt->bindParam(":id",$data, PDO::PARAM_INT);


    $stmt->execute();


    return $stmt->fetch();


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

El modelo **editarUnidadModel:** obtiene la información de una unidad en especifico. Una vez obtenida esta información, la retorna al controlador para ser mostrada y editada.

```
public static function modificarUnidadModel($data,$tabla)
{
    $stmt = Conexion::conectar()->prepare("UPDATE $tabla SET nombre = :nombre,
    fecha_inicio = :inicio, fecha_fin = :fin WHERE id_unidad = :id");

    $stmt -> bindParam(":nombre",$data["nombre"],PDO::PARAM_STR);
    $stmt -> bindParam(":inicio",$data["inicio"],PDO::PARAM_STR);
    $stmt -> bindParam(":fin",$data["fin"],PDO::PARAM_STR);
    $stmt -> bindParam(":id",$data["id"],PDO::PARAM_INT);

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

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

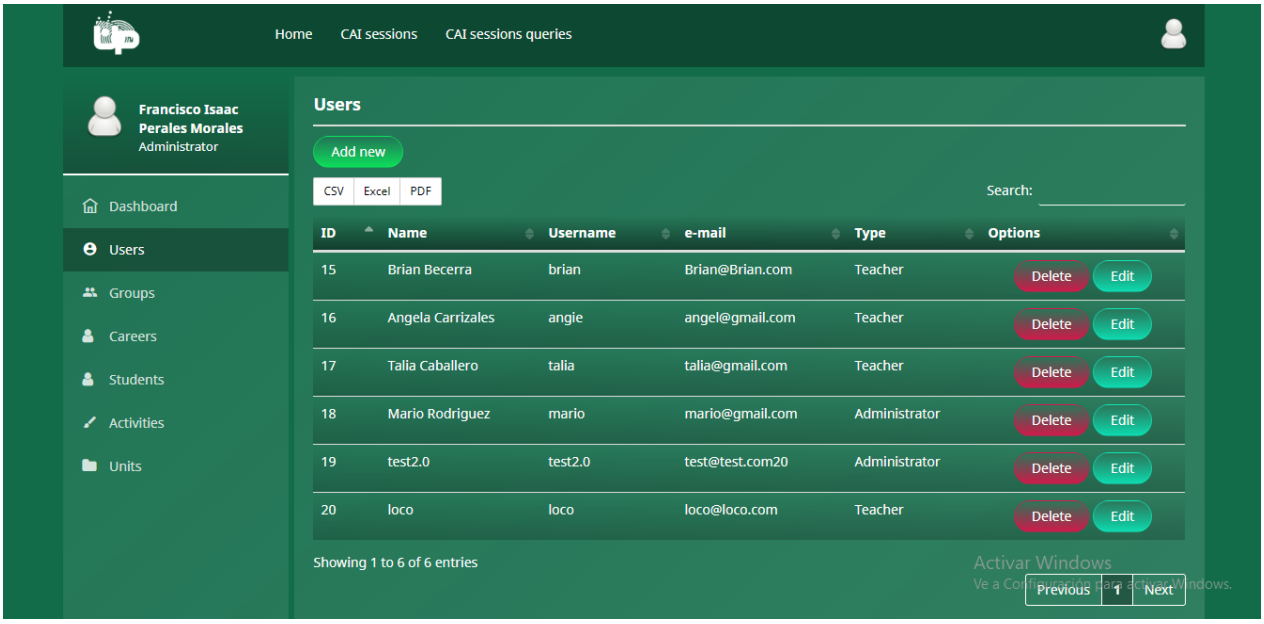
El modelo **modificarUnidadModel:** este modelo recibe la información modificada de una unidad, una vez recibida esta información, actualiza la información existente en la base de datos de la unidad, con la información recibida.



```
    public static function optionUnidadModel($tabla)
    {
        $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla");

        $stmt -> execute();

        return $stmt -> fetchAll();

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

Por ultimo esta el modelo **optionUnidadModel:** este modelo obtiene la información de las unidades registradas en la base de datos. una vez que obtiene la información de las unidades, la retorna al controlador para que pueda ser mostrada en un select.

MÓDULO DE USUARIOS



CONTROLADOR DE USUARIOS(controllerUsuario.php)

```php
<?php

class mvcUsuario
{

    function agregarUsuarioController()
    {

        if(isset($_POST["nombre"]))
        {

            $data = array("nombre" => $_POST["nombre"],
                          "username" => $_POST["username"],
                          "password" => $_POST["password"],
                          "email" => $_POST["email"],
                          "tipo" => $_POST["tipo"]);


            $resp = CRUDUsuario::agregarUsuarioModel($data,"usuario","teacher");


            if($resp == "success")
            {

                $_SESSION["mensaje"] = "add";


                echo "<script>
                        window.location.replace('index.php?section=users&action=list');
                      </script>";
            }
            else
            {

                echo "error";
            }
        }
    }
```

El primer controlador que contiene es **agregarUsuarioController**, este controlador se encarga de recibir la información de un nuevo usuario a registrar en ella sistema. Una vez recibida la información la almacena en un array y la enviá a agregarUsuarioModel para que sea registrada en la base datos, en caso de que se haya registrado exitosamente nos redireccionara al listado de usuarios.

```
function listadoUsuarioController()
{

    $data = CRUDUsuario::listadoUsuarioModel("usuario");


    foreach($data as $rows => $row)
    {

        echo "<tr class='fondoTabla'>
            <td>".$row["num_empleado"]."</td>
            <td>".$row["nombre"]."</td>
            <td>".$row["username"]."</td>
            <td>".$row["email"]."</td>
            <td>".$row["tipo"]."</td>
            <td>
                <center>
                    <button class='btn btn-rounded btn-danger' data-toggle='modal' data-
                    target='=delete-modal' onclick='idDel(".$row["num_empleado"].")'> Delete
                    </button>

                    <a href='index.php?
                    section=users&action=list&edit=".$row["num_empleado"]."'>
                        <button class='btn btn-rounded btn-custom'>Edit</button>
                    </a>
                </center>
            </td>
            </tr>";
    }
}
```

El controlador **listadoUsuarioController:** este controlador recibe la información de los usuarios registrados mediante listadoUsuarioModel, para así imprimir su información en pantalla para ser mostrada mediante una tabla.

```php
public function editarUsuarioController()
{

    $data = $_GET["edit"];


    $resp = CRUDUsuario::editarUsuarioModel($data,"usuario");


    echo
    "
                <input type='hidden' name='id' value=".$data.">
                <div class='form-group'>
                    <label class='control-label repairtext'>Name</label>
                    <input type='text' class='form-control' name='nombre' placeholder='Name'
                    value='".$resp["nombre"]."' required>
                </div>

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


                <div class='form-group'>
                    <label class='control-label repairtext'>Password</label>
                    <input type='password' class='form-control' name='password'
                    placeholder='Password' value='".$resp["password"]."' required>
                </div>


                <div class='form-group'>
                    <label class='control-label repairtext'>Email</label>
                    <input type='email' class='form-control' name='email'
                    placeholder='Email' value='".$resp["email"]."'>
                </div>
    ";
}
```

El controlador **editarUsuarioController:** este controlador muestra la información de un usuario para ser modificada, para esto recibe del sistema el id del usuario y se lo enviá a editarUsuarioModel para obtener la información del usuario y después lo imprima su información para ser mostrada en un formulario.

```php
public function eliminarUsuarioController()
{

    if(isset($_POST["del"]))
    {

        $data = $_POST["del"];


        $resp = CRUDUsuario::eliminarUsuarioModel($data,"grupo","teacher","usuario");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "delete";


            echo "<script>
                    window.location.replace('index.php?section=users&action=list');
                  </script>";
        }
    }
}
```

El controlador **eliminarUsuarioController:** este controlador recibe del sistema el id del usuario al cual se eliminara de la base de datos, para esto se lo enviá a eliminarUsuarioModel para que borre su información de la base de datos, en caso de que se el usuario se haya borrado correctamente, nos redireccionara al listado de usuarios.

```php
public function modificarUsuarioController()
{


    if(isset($_POST["id"]))
    {

        $data = array("id" => $_POST["id"],
                      "nombre" => $_POST["nombre"],
                      "username" => $_POST["username"],
                      "password" => $_POST["password"],
                      "email" => $_POST["email"]);


        $resp = CRUDUsuario::modificarUsuarioModel($data,"usuario");


        if($resp == "success")
        {

            $_SESSION["mensaje"] = "edit";


            echo "<script>
                    window.location.replace('index.php?section=users&action=list');
                  </script>";
        }
    }
}
```

El controlador **modificarUsuarioController:** este controlador recibe del sistema la información modificada de un usuario, esta información la guarda un array y después se lo manda a modificarUsuarioModel para que actualice su información en la base de datos, en caso de que la información del usuario se haya actualizado correctamente, nos redireccionara al listado de usuarios.

```
    public function optionUsuarioController()
    {

        $data = CRUDUsuario::optionUsuarioModel("usuario","teacher");


        foreach($data as $rows => $row)
        {

            echo "<option value=".$row["num_empleado"].">".$row["nombre"]."</option>";
        }
    }
}
?>
```

Por ultimo esta **optionUsuarioController:** la función de este controlador es mostrar la información de los usuarios que son Teacher en un Select. Para esto recibe la información de los teacher de optionUsuarioModel y después imprime la información, para que sea mostrada en un Select.

CRUD DE USUARIOS(crudUsuario.php)

Este archivo contiene la clase de **CRUDUsuario**, esta clase contiene los modelos para manejar las operaciones en la base de datos para la sección de usuarios del sistema.

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


class CRUDUsuario
{

    public static function agregarUsuarioModel($data,$tabla1,$tabla2)
    {

        $stmt = Conexion::conectar() -> prepare("INSERT INTO $tabla1
        (nombre,username,password,email,tipo) VALUES
        (:nombre,:username,:password,:email,:tipo)");
```

```
        $stmt -> bindParam(":nombre",$data["nombre"],PDO::PARAM_STR);
        $stmt -> bindParam(":username",$data["username"],PDO::PARAM_STR);
        $stmt -> bindParam(":password",$data["password"],PDO::PARAM_STR);
        $stmt -> bindParam(":email",$data["email"],PDO::PARAM_STR);
        $stmt -> bindParam(":tipo",$data["tipo"],PDO::PARAM_STR);


        if($stmt -> execute())
        {

            if($data["tipo"] == "Teacher")
            {

                $stmt = Conexion::conectar() -> prepare("SELECT MAX(num_empleado) as id
                FROM $tabla1");

                $stmt -> execute();

                $id = $stmt -> fetch();

                $id = $id["id"];
```

El primer modelo de la clase es **agregarUsuarioModel:** este modelo recibe la información de un nuevo usuario el cual se registra en la base de datos. Dependiendo del tipo de usuario que sea se registra en uno o 2 tablas esta información.

```
public static function listadoUsuarioModel($tabla)
{

    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla WHERE NOT num_empleado = :id");
    $stmt -> bindParam(":id",$_SESSION["empleado"],PDO::PARAM_INT);


    $stmt -> execute();


    return $stmt -> fetchAll();


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

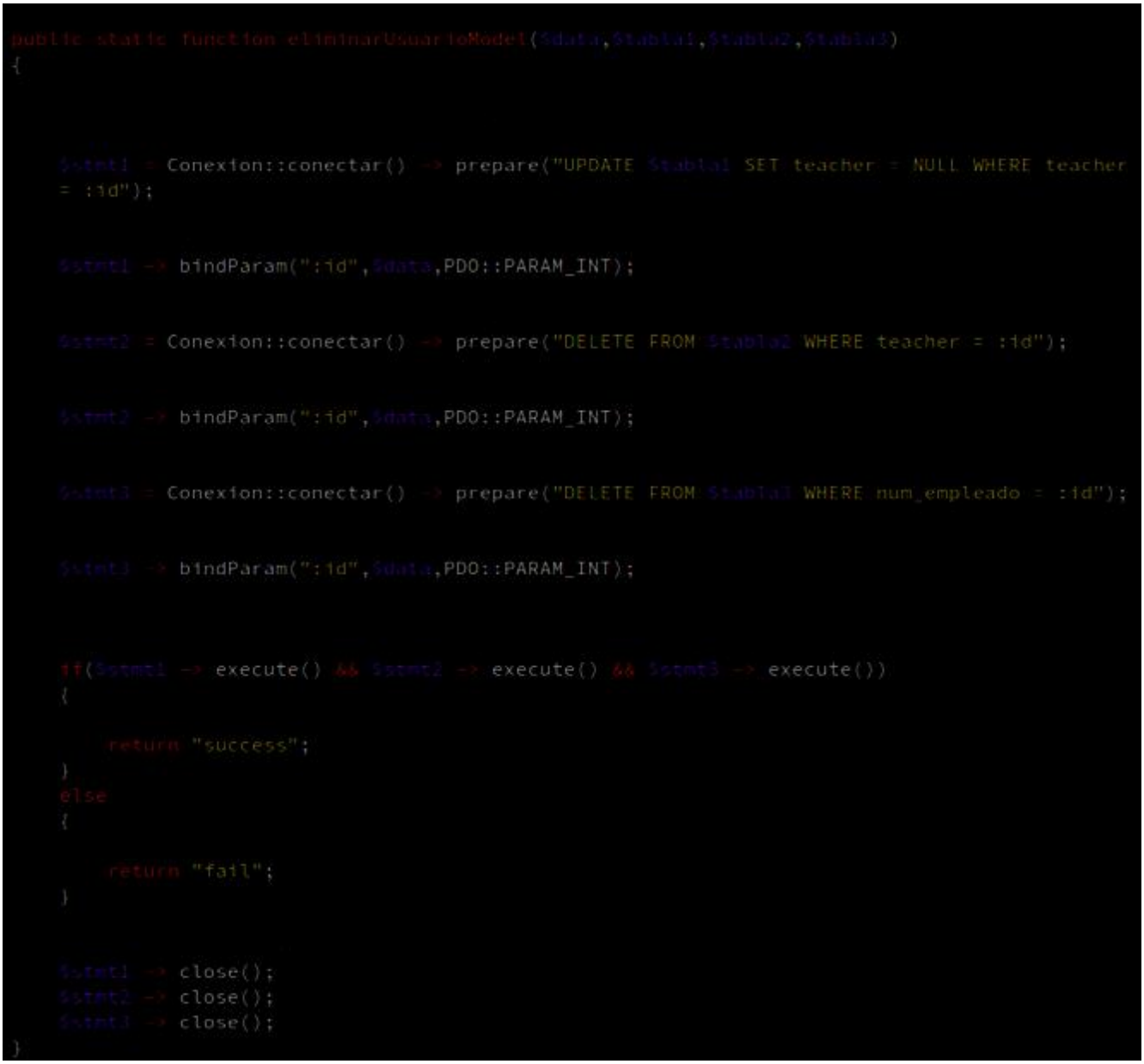
El modelo **listadoUsuarioModel:** este modelo obtienen la información de los usuarios registrados en el sistema, para después ser retornada al controlador para ser mostrada.

```php
public static function optionUsuarioModel($tabla1,$tabla2)
{

    $stmt = Conexion::conectar() -> prepare("SELECT * FROM $tabla1 as u JOIN $tabla2 as t on
    t.teacher = u.num_empleado");


    $stmt -> execute();


    return $stmt -> fetchAll();


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

El modelo **optionUsuarioModel:** este modelo obtienen la información de los usuarios registrados en el sistema, para después ser retornada al controlador para ser mostrada en un select.

```php
public static function editarUsuarioModel($data,$tabla)
{

    $stmt = Conexion::conectar()->prepare("SELECT * FROM $tabla WHERE num_empleado = :id");


    $stmt->bindParam(":id",$data, PDO::PARAM_INT);


    $stmt->execute();


    return $stmt->fetch();


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

El modelo **editarUsuarioModel:** este modelo obtiene la información de un usuario en especifica, para que sea mostrada y después editada.



```
public static function eliminarUsuarioModel($data,$tabla1,$tabla2,$tabla3)
{

    $stmt1 = Conexion::conectar()->prepare("UPDATE $tabla1 SET teacher = NULL WHERE teacher
    = :id");

    $stmt1->bindParam(":id",$data,PDO::PARAM_INT);

    $stmt2 = Conexion::conectar()->prepare("DELETE FROM $tabla2 WHERE teacher = :id");

    $stmt2->bindParam(":id",$data,PDO::PARAM_INT);

    $stmt3 = Conexion::conectar()->prepare("DELETE FROM $tabla3 WHERE num_empleado = :id");

    $stmt3->bindParam(":id",$data,PDO::PARAM_INT);


    if($stmt1->execute() && $stmt2->execute() && $stmt3->execute())
    {

        return "success";
    }
    else
    {

        return "fail";
    }


    $stmt1->close();
    $stmt2->close();
    $stmt3->close();
}
```

El modelo **eliminarUsuarioController:** este modelo se encarga de el borrado de la información de un usuario y aquella con la cual también esta relacionada la información del usuario.