

**SISTEMA DE CAI (CENTRO DE APRENDIZAJE DE IDIOMAS)**  
**MANUAL TECNICO**

UNIVERSIDAD POLITECNICA DE VICTORIA  
INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
TECNOLOGIAS Y APLICACIONES WEB  
M.S.I. MARIO HUMBERTO RODRIGUEZ CHAVEZ

INTEGRANTES DE EQUIPO:  
GOMEZ FLORES OSIEL  
HERNANDEZ RODRIGUEZ MIGUEL ANGEL

CD. VICTORIA, TAMAULIPAS. MEXICO

El sistema esta creado con el Modelo Vista Controlador (MVC).

Utilizando el lenguaje de programacion web del lado del servidor, PHP.

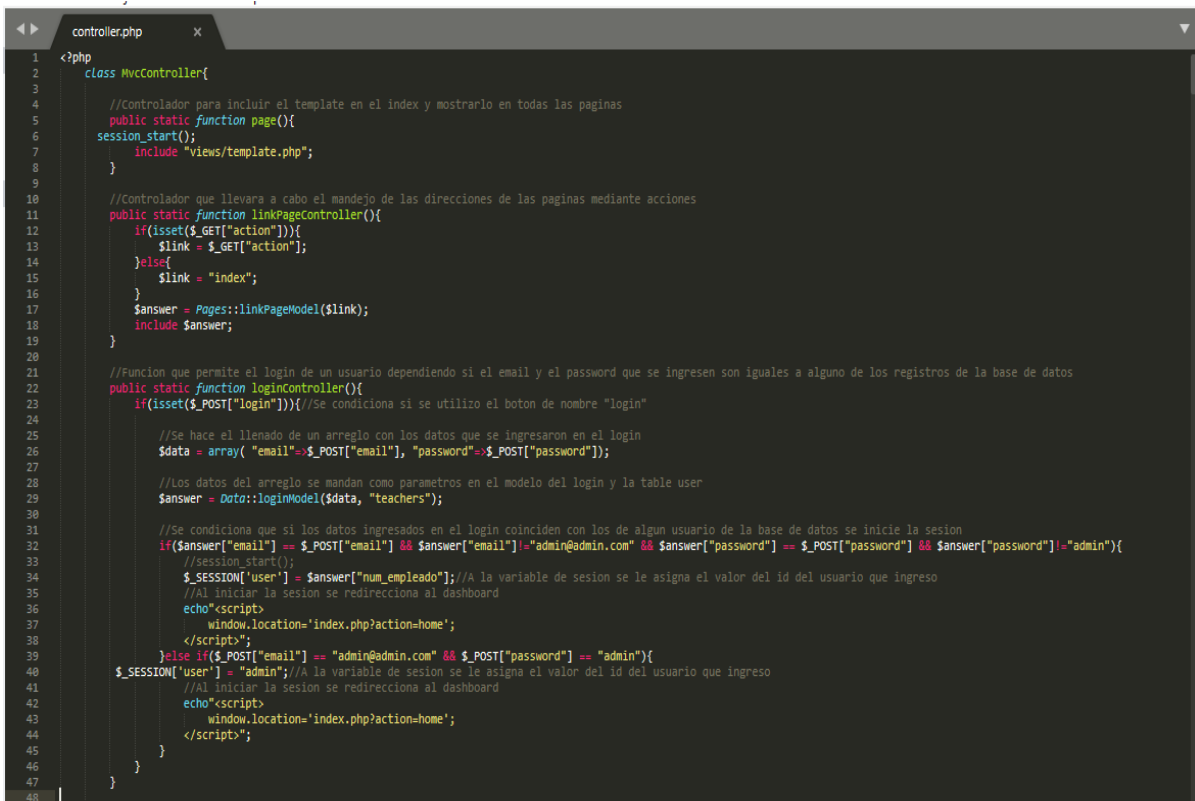
Lenguaje de marcado que es HTML.

Lenguaje del lado del cliente JavaScript.

Utilizando el material de diseño Materialize.

Comenzando con la clase principal la cual es la clase “MvcController”, en dicha clase se encuentran todas las funciones (controladores) que se utilizan para que el sistema tenga un correcto funcionamiento

## CONTROLADOR.



```
1 <?php
2 class MvcController{
3
4     //Controlador para incluir el template en el index y mostrarlo en todas las paginas
5     public static function page(){
6         session_start();
7         include "views/template.php";
8     }
9
10    //Controlador que llevara a cabo el manejo de las direcciones de las paginas mediante acciones
11    public static function linkPageController(){
12        if(isset($_GET["action"])){
13            $link = $_GET["action"];
14        }else{
15            $link = "index";
16        }
17        $answer = Pages::linkPageModel($link);
18        include $answer;
19    }
20
21    //Funcion que permite el login de un usuario dependiendo si el email y el password que se ingresen son iguales a alguno de los registros de la base de datos
22    public static function loginController(){
23        if(isset($_POST["login"])){//Se condiciona si se utilizo el boton de nombre "login"
24
25            //Se hace el llenado de un arreglo con los datos que se ingresaron en el login
26            $data = array( "email"=>$_POST["email"], "password"=>$_POST["password"]);
27
28            //Los datos del arreglo se mandan como parametros en el modelo del login y la table user
29            $answer = Data::loginModel($data, "teachers");
30
31            //Se condiciona que si los datos ingresados en el login coinciden con los de algun usuario de la base de datos se inicie la sesion
32            if($answer["email"] == $_POST["email"] && $answer["email"]!="admin@admin.com" && $answer["password"] == $_POST["password"] && $answer["password"]!="admin"){
33                //session start();
34                $_SESSION["user"] = $answer["num_empleado"]; //A la variable de sesion se le asigna el valor del id del usuario que ingreso
35                //Al iniciar la sesion se redirecciona al dashboard
36                echo<script>
37                    window.location="index.php?action=home";
38                </script>;
39            }else if($_POST["email"] == "admin@admin.com" && $_POST["password"] == "admin"){
40                $_SESSION["user"] = "admin"; //A la variable de sesion se le asigna el valor del id del usuario que ingreso
41                //Al iniciar la sesion se redirecciona al dashboard
42                echo<script>
43                    window.location="index.php?action=home";
44                </script>;
45            }
46        }
47    }
48 }
```

En esta captura hay 3 controladores: page(), linkPageController() y LoginController().

page: este controlador incluye donde sea llamado, la vista “template.php” mediante un “include()”.

linkPageController: este controlador obtiene mediante un “\$\_GET[“action”]”, esto se obtiene de la url, obtiene el valor del “action” y lo asigna a una variable “\$link”, si no existe “action” en la url, entonces se asigna “index” a la variable, y esa variable es mandada a un método del modelo el cual re-direcciona a una vista determinada.

loginController: este controlador realiza la función de iniciar sesión, si el botón “login” del formulario fue presionado o lanzado, esto lo hace mediante la obtención de los valores que el usuario llenó en el formulario de la vista de login, los cuales son el nombre de usuario y contraseña y los almacenamos en un array, y este array es mandado a un método del modelo donde se realizan todas las consultas a la base de datos, aquí se usa el método “Data::loginModel()” la cual recibe 2 parámetros, el array con el nombre de usuario y contraseña y el nombre de la tabla en la cual va a hacer la consulta, este método hace consulta en la tabla “teachers” y devuelve los datos del teacher donde coincidan los valores enviados, si los valores devueltos son iguales a los valores ingresados, es decir, si los datos coincidieron, entonces quiere decir que el usuario existe, por lo cual se inicia una variable “\$\_SESSION[“user”]” y le asignamos el número de empleado del teacher y re-direcciona a la página principal del sitio poniendo un “action=home”. Si la consulta que devolvió el modelo no coincidió con los datos ingresados por el usuario, el sitio re-direcciona a la misma página de login.

```
controller.php  crud.php
56
57 public static function showTeachersController(){
58     $answer = Data::showTeachersModel("teachers");
59     foreach ($answer as $row => $teacher) {
60
61         echo'
62         <tr>
63             <td>'.$teacher['num_empleado'].'</td>
64             <td>'.$teacher['nombre_teacher'].'</td>
65             <td>'.$teacher['apellidos_teacher'].'</td>
66             <td>'.$teacher['email'].'</td>';
67
68             if($teacher['foto']=="views/img/" || $teacher['foto']==""){
69                 echo'<td><button class="btn" disabled><i class="material-icons">person_outline</i></button></td>';
70             }else{
71                 echo'<td></td>';
72             }
73         echo ' <td>'.$teacher['telefono'].'</td>
74         <td><a class="btn-floating btn-large waves-effect waves-light teal darken-3 btn modal-trigger" href="#"><i
75             class="material-icons">edit</i></a></td>
76         <td><a class="btn-floating btn-large waves-effect waves-light red darken-4 btn modal-trigger" href="#"><i
77             class="material-icons">delete</i></a></td>
78         </tr>
79         <div id="delete_'.$teacher['num_empleado'].'" class="modal modal-fixed-footer" style="height:270px">
80             <div class="modal-content">
81                 <h4>Are you sure you want to delete this teacher?</h4><br>
82                 <button class="waves-effect waves-light red darken-4 btn-large modal-close">No</button>
83                 <a class="waves-effect waves-light blue darken-4 btn-large" href="index.php?action=eliminarTeacher&num_empleado='.$teacher['num_empleado'].'">Yes</a>
84             </div>
85         </div>
86     ';
87 }
88 echo "</tbody>";
89 </table>";
90 foreach ($answer as $row => $teacher) {
91     $ucController::editTeacherController($teacher["num_empleado"]);
92 }
93 }
94
95 public static function addTeacherController(){
96     if(isset($_POST["add"])){
97         $target_dir = "views/img/";
98         $target_file = $target_dir . basename($_FILES["foto"]["name"]);
99         $uploadOk = 1;
```

ShowTeacherController: este controlador primeramente hace una consulta mediante el modelo “data::showTeacherModel” de la tabla “teachers” y almacena el resultado en una variable y después este resultado se imprime en una tabla con los resultados imprimiendo todos los campos de la tabla “teachers”, poniendo las facilidades para borrar y modificar, utilizando modales.

```

94 }
95
96 public static function addTeacherController(){
97     if(isset($_POST['add'])){
98
99         $target_dir = "views/img/";
100         $target_file = $target_dir . basename($_FILES["foto"]["name"]);
101         $uploadOk = 1;
102         // Check if file already exists
103         if (move_uploaded_file($_FILES["foto"]["tmp_name"], $target_file)) {
104             $ruta = basename($_FILES["foto"]["name"]);
105             //echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
106         } else {
107             echo "Sorry, there was an error uploading your file.";
108         }
109
110         $data = array("num_empleado"=>$_POST["num_empleado"], "nombre_teacher"=>$_POST["nombre_teacher"], "apellidos_teacher"=>$_POST["apellidos_teacher"], "email"=>$_POST["email"], "password"=>$_POST["password"], "foto"=>$target_file, "telefono"=>$_POST["telefono"]);
111         $answer = Data::addTeacherModel($data, "teachers");
112         var_dump($answer);
113         if($answer=="success"){
114             echo "<script>
115                 window.location='index.php?action=teachers';
116             </script>";
117         }else{
118             echo "Error al registrar";
119         }
120     }
121 }
122
123
124
125
126
127 public static function deleteTeacherController(){
128     $data = $_GET["num_empleado"];
129     $answer = Data::deleteTeacherModel($data, "teachers");
130     if($answer == "success"){
131         echo "<script>
132             window.location='index.php?action=teachers';
133         </script>";
134     }else{
135         echo "Error al Eliminar";
136     }
137 }
138
139 public static function editTeacherController($data){
140     $answer = Data::editTeacherModel($data, "teachers");

```

**addTeacherController:** este controlador hace la funcionalidad de agregar un nuevo teacher a la base de datos. Primeramente verifica si el botón “add” fue presionado para poder almacenar los datos del formulario llenado por el usuario, primeramente, almacena los datos de la imagen y la mueve a la carpeta de destino para las imágenes, después almacena en un array todos los datos del formulario como numero de emp,leado, nombre, apellidos, etc., y manda a un método del modelo “data::addTeacherModel”, el array con esos datos y el nombre de la tabla a insertar.

Según la respuesta del método del modelo, si es “success” entonces re-direcciona a la vista de “teachers” poniendo un “action” en la url de “teachers”, si no es “success” la respuesta del modelo, entonces muestra un mensaje de error en el registro del teacher.

**deleteTeacherController:** este controlador almacena en una variable un valor de la url obtenido por un “\$\_GET[“num\_empleado”]” y esta variable se manda a un método del modelo llamado “data::deleteTeacherModel” junto con el nombre de la tabla de la base de datos par eliminar el teacher, la tabla “teachers” y según la respuesta del modelo, re-direcciona a la vista “teachers”

```

139 public static function editTeacherController($data){
140     $answer = Data::editTeacherModel($data,"teachers");
141     echo'
142     <form method="post" enctype="multipart/form-data">
143     <div id="'. $answer['num_empleado'].'" class="modal modal-fixed-footer">
144     <div class="modal-content">
145     <h4>Info</h4>
146     <div class="row">
147     <div class="input-field col s12">
148     <input type="hidden" name="num_empleado" value="'. $answer['num_empleado'].'">
149     </div>
150     <div class="input-field col s6">
151     <input type="text" name="nombre_teacher" value="'. $answer['nombre_teacher'].'">
152     <label for="nombre_teacher">Name</label>
153     </div>
154     <div class="input-field col s6">
155     <input type="text" name="apellidos_teacher" value="'. $answer['apellidos_teacher'].'">
156     <label for="apellidos_teacher">Last Name</label>
157     </div>
158     <div class="input-field col s6">
159     <input type="email" name="email" value="'. $answer['email'].'">
160     <label for="email">E-Mail</label>
161     </div>
162     <div class="input-field col s6">
163     <input type="password" name="password" value="'. $answer['password'].'">
164     <label for="password">Password</label>
165     </div>
166     <div class="input-field col s7">
167     <label>Photo</label><br><br>
168     <input type="file" name="foto" class="waves-effect waves-light btn-small">
169     </div>
170     <div class="input-field">
171     <input type="hidden" name="fotoActual" value="'. $answer['foto'].'">
172     </div>
173     <div class="col s5"></div>
174     <div id="verImagenEdit" class="col s5" style="width: auto; height: auto;"></div>
175     <div class="input-field col s5">
176     <input type="text" name="telefono" value="'. $answer['telefono'].'">
177     <label for="telefono">Phone</label>
178     </div>
179     </div>
180     </div>
181     <div class="modal-footer">
182     <button type="submit" class="waves-effect waves-light blue darken-4 btn-small" name="update">Update Information</button>
183     </div>
184     </div>
185     </div>

```

editTeacherController: este controlador recibe como parámetro el número de empleado, es decir el ID del teacher para posteriormente realizar una consulta mediante el método del modelo "Data::editTeacherModel" que recibe como parámetros, el ID del teacher que llega mediante la función y el nombre de la tabla a consultar "teachers", y este modelo nos devuelve todos los datos de ese teacher y los almacena en una variable "answer", esta variable es utilizada para mostrar todos los campos del teacher y ver su imagen también, y finalmente hay un botón "update information" el cual actualizará los campos en la tabla "teachers" de la base de datos y todo esto es mostrado en un modal.

```

190     }
191
192     public static function updateTeacherController(){
193         if(isset($_POST["update"])){
194             if($_FILES["foto"]["name"]!=null){
195                 $target_dir = "views/img/";
196                 $target_file = $target_dir . basename($_FILES["foto"]["name"]);
197                 $uploadOk = 1;
198                 // Check if file already exists
199                 if (move_uploaded_file($_FILES["foto"]["tmp_name"], $target_file)) {
200                     $ruta = basename($_FILES["foto"]["name"]);
201                     //echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
202                 } else {
203                     echo "Sorry, there was an error uploading your file.";
204                 }
205             }else{
206                 $target_file = $_POST["fotoActual"];
207             }
208
209
210             $data = array("nombre_teacher"=>$_POST["nombre_teacher"],"apellidos_teacher"=>$_POST["apellidos_teacher"],"email"=>$_POST["email"],"password"=>$_POST["password"], "foto"=>$target_file, "telefono"=>$_POST["telefono"],"num_empleado"=>$_POST["num_empleado"]);
211
212             $answer = Data::updateTeacherModel($data, "teachers");
213
214             if($answer == "success"){
215                 echo "<script>
216                     window.location='index.php?action=teachers';
217                 </script>";
218             }
219             else{
220                 echo "Error al Actualizar";
221             }
222             var_dump($data);
223         }
224     }
225
226 }
227
228
229 #####
230 #####

```

updateTeacherController: este es el controlador que es llamado al dar clic en “update information” del modal para modificar un teacher, en el cual este controlador primeramente verifica si la imagen también fue cambiada o se quedó igual para guardar la imagen moviéndola a la carpeta de destino o dejar la misma que ya tenía guardada, y si existe un error al mover la imagen a la carpeta de destino se imprime un error. Posteriormente se almacenan todos los demás campos llenados en el formulario para modificar los datos de un teacher específico, se obtienen los datos mediante “\$\_POST[“”]” del teacher y se almacenan en un variable “array” para después mandarla con el nombre de la tabla “teachers” a un método del modelo llamado “Data::updateTeacherModel()” para que este realice una actualización de datos de ese teacher específico en la base de datos, si la actualización de datos se realizó correctamente se re-direcciona al usuario a la vista “teachers” poniéndolo en la url con un “action”, si no, entonces imprime un mensaje de error al actualizar los datos.

```

409 controller.php x crud.php x connection.php x
405 //este metodo utilizara los datos que se encuentren dentro del metodo editGroupController, ya que son lo que se utilizaran para hacer la modificacion del grupo
406 public static function updateGroupController(){
407     if(isset($_POST["update"])){
408
409         $data = array("id_grupo"=>$_POST["id_grupo"],"nombre_grupo"=>$_POST["nombre_grupo"],"teacher"=>$_POST["teacher"]);
410         $answer = Data::updateGroupModel($data, "grupos");
411         if($answer == "success"){
412             echo "<script>
413                 window.location='index.php?action=grupos';
414             </script>";
415         }
416         else{
417             echo "Error al Actualizar";
418         }
419     }
420 }
421
422 #####
423 ##### STUDENTS CONTROLLER #####
424 #####
425
426 public static function showStudentsController(){
427     $answer = Data::showStudentsModel("alumnos","grupos");
428     foreach ($answer as $row => $student) {
429         echo'
430         <tr>
431             <td>'.$student['matricula'].'</td>
432             <td>'.$student['nombre_alumno'].'</td>
433             <td>'.$student['apellidos_alumno'].'</td>
434             <td>'.$student['email'].'</td>
435             <td>'.$student['carrera'].'</td>
436             <td>'.$student['nombre_grupo'].' - Level '.$student['nivel'].'</td>
437             <td>'.$student['nombre_teacher'].' '.$student['apellidos_teacher'].'</td>
438
439             <td><a class="btn-floating btn-large waves-effect waves-light teal darken-3 btn modal-trigger" href="#">'.$student["matricula"].'<i
440                 class="material-icons">edit</i></a></td>
441             <td><a class="btn-floating btn-large waves-effect waves-light red darken-4 btn modal-trigger" href="#delete'.$student["matricula"].'><i
442                 class="material-icons">delete</i></a></td>
443         </tr>
444
445         <div id="delete'.$student["matricula"].'" class="modal modal-fixed-footer" style="height:270px">

```

updateGroupController: este controlador es lanzado cuando se presiona el botón “update”, y primeramente obtiene los datos del grupo llenados en el formulario mediante “\$\_POST[]” y estos datos los almacenamos en una variable “\$data” que almacenará un array con todos esos datos, posteriormente hace uso de un método del modelo llamado “Data::updateGroupModel()” al cual se le mandan 2 parámetros, el primero es el array con todos los datos del grupo y el segundo es el nombre de la tabla a actualizar “grupos”, si la respuesta devuelta por el modelo es “success” fue exitosa y re-direcciona a la vista “grupos” poniendo un “action” en la url, de lo contrario se mostrará un mensaje de error al actualizar.

```

345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386

amprimaren dichos datos para poder visualizarlos y poder editarlos
public static function editGroupController($data){
    $answer = Data::editGroupModel($data,"grupos","teachers");
    echo '
    <form method="post" enctype="multipart/form-data">
    <div id="'. $answer["id_grupo"].'" class="modal modal-fixed-footer">
    <div class="modal-content">
    <h4>Info</h4>
    <div class="row">
    <div class="input-field col s12">
    <input type="hidden" name="id_grupo" value="'. $answer['id_grupo'].'">
    </div>
    <div class="input-field col s12">
    <input type="text" name="nombre_grupo" value="'. $answer['nombre_grupo'].'">
    <label for="nombre_grupo">Name</label>
    </div>
    <div class="input-field col s6">
    <select name="nivel">
    <optgroup label="Current Level">
    <option value="'. $answer["nivel"].'">'. $answer["nivel"].'</option>
    </optgroup>
    <optgroup label="All Levels">
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
    <option value="9">9</option>
    </optgroup>
    </select>
    <label>Level</label>
    </div>
    <div class="input-field col s6">
    <select name="teacher">
    <optgroup label="Current Teacher">
    <option value="'. $answer["num_empleado"].'">'. $answer["nombre_teacher"].'</option>
    </optgroup>
    </select>
    </div>
    </div>
    </div>
    </form>
    ';
```

editGroupController: este controlador recibe como parámetro el id del grupo indicado para editar, primeramente se utiliza un método del modelo llamado "Data::editGroupModel()" el cual recibe 3 parámetros, el primero es el id del grupo, el segundo es el nombre de la tabla grupos, y el tercero es el nombre de la tabla teachers y el modelo devuelve en un array los campos del id del grupo y los datos del teacher donde coincida con ese grupo. Después procede a mostrar los resultados de la consulta en un modal, imprimiendo los datos del grupo como id de grupo, nombre del grupo, nivel, y en esta parte donde imprime el nivel, se puso un "optgroup" el cual equivale a un select, en el cual se muestran todos los niveles de inglés posibles, y también te muestra los datos del teacher actual de ese grupo, y todos los datos se colocan en "input" disponibles para ser modificados y al final puedes dar clic al botón para actualizar y procederá a guardar los datos que estén en los "input" para actualizarlos en la base de datos.



```

488
489
490 public static function editStudentController($data){
491     $answer = Data::editStudentModel($data,"alumnos","grupos");
492     echo'
493     <form method="post" enctype="multipart/form-data">
494     <div id="'.$answer["matricula"].'" class="modal modal-fixed-footer">
495     <div class="modal-content">
496     <h4>Info</h4>
497     <div class="row">
498     <div class="input-field col s12">
499     <input type="hidden" name="matricula" value="'.$answer["matricula"].'">
500     </div>
501     <div class="input-field col s6">
502     <input type="text" name="nombre_alumno" value="'.$answer["nombre_alumno"].'">
503     <label for="nombre_grupo">Name</label>
504     </div>
505     <div class="input-field col s6">
506     <input type="text" name="apellidos_alumno" value="'.$answer["apellidos_alumno"].'">
507     <label for="apellidos_alumno">Last Name</label>
508     </div>
509     <div class="input-field col s6">
510     <select name="carrera">
511     <optgroup label="Current Carrer">
512     <option value="'.$answer["carrera"].'">'.$answer["carrera"].'</option>
513     </optgroup>
514     <optgroup label="All Carrers">
515     <option value="Ingeniería en Tecnologías de la Informacion">Ingeniería en Tecnologías de la Informacion</option>
516     <option value="Ingeniería en Tecnologías de la Manufactura">Ingeniería en Tecnologías de la Manufactura</option>
517     <option value="Ingeniería en Mecatronica">Ingeniería en Mecatronica</option>
518     <option value="Ingeniería en Sistemas Automotries">Ingeniería en Sistemas Automotries</option>
519     <option value="Licenciatura en Gestion y Administracion de Pequeñas y Medianas Empresas">Licenciatura en Gestion y
520     Administracion de Pequeñas y Medianas Empresas</option>
521     </optgroup>
522     </select>
523     </div>
524     <div class="input-field col s6">
525     <select name="grupo">
526     <optgroup label="Current Group">
527     <option value="'.$answer["grupo"].'">'.$answer["grupo"].'</option>
528     </optgroup>
529     <optgroup label="All Groups">
530     <option value="Grupo 1">Grupo 1</option>
531     <option value="Grupo 2">Grupo 2</option>
532     <option value="Grupo 3">Grupo 3</option>
533     <option value="Grupo 4">Grupo 4</option>
534     <option value="Grupo 5">Grupo 5</option>
535     </optgroup>
536     </select>
537     </div>
538     </div>
539     </div>
540     </form>
541     ';
```

editStudentController(): este controlador recibe como parámetro la matrícula del alumno a editar, y realiza una consulta a la base de datos mediante el modelo "Data::editStudentModelo()") mandando como parámetro el id del estudiante, la tabla a consultar de la base de datos "alumnos" y el nombre de la tabla "grupos", y este modelo devuelve todos los datos del alumno y su grupo, y es almacenado en una variable "answer" para después mostrar todos esos datos obtenidos mediante el modelo, en un modal, imprimiendo todos los campos del alumno, como su matrícula, nombre, apellidos y carrera, en este campo carrera se mostrará la que actualmente existe en el registro de la base de datos, pero también se le muestra una lista de posibles carreras a seleccionar ya que esta parte es para editar donde posiblemente puedes cambiarlo de carrera, y al final dando clic al botón podremos actualizar los datos del alumno en la base de datos.

```

456     }
457 }
458 }
459
460 //Se ejecuta cuando se realiza un submit y un post, al momento de recibir la variable "add" mediante el metodo post, se reciben todas las variables del
461 formulario de agregar a un estudiante
462 public static function addStudentController(){
463     if(isset($_POST["add"])){
464         $data = array("matricula"=>$_POST["matricula"], "nombre_alumno"=>$_POST["nombre_alumno"], "apellidos_alumno"=>$_POST["apellidos_alumno"], "carrera"=>$_
465             POST["carrera"], "grupo"=>$_POST["grupo"], "email"=>$_POST["email"]);
466         $answer = Data::addStudentModel($data, "alumnos");
467         if($answer=="success"){
468             echo "<script>
469                 window.location='index.php?action=alumnos';
470             </script>";
471         }else{
472             var_dump($data);
473             echo "Error al registrar";
474         }
475     }
476 }
477
478 public static function deleteStudentController(){
479     $matricula = $_GET["matricula"];
480     $answer = Data::deleteStudentModel($matricula, "alumnos");
481     if($answer == "success"){
482         echo "<script>
483             window.location='index.php?action=alumnos';
484         </script>";
485     }else{
486         echo "Error al eliminar";
487     }
488 }
489
490 public static function editStudentController($data){
491     $answer = Data::editStudentModel($data, "alumnos", "grupos");
492     echo "
493     <form method='post' enctype='multipart/form-data'>
494     <div id=''. $answer['matricula']. ' ' class='modal modal-fixed-footer'>
495     <div class='modal-content'>
496     <h4>Info</h4>

```

addStudentController: este controler verifica si fue presionado el botón “add” para agregar un nuevo alumno o estudiante de inglés, si se presionó el botón entonces procede a obtener todos los datos llenados sobre el alumno en el formulario de llenado de alumno nuevo mediante el uso de “\$\_POST[]”, como su nombre, apellidos y más, guardándolos en un array llamado “\$data”, después se usa el modelo “Data::addStudentModel()” en el cual se le manda como parámetros el array con los datos del alumno y el nombre de la tabla a insertar los datos “alumnos”, si la inserción en la tabla fue exitosa entonces se re-direcciona a la vista “alumnos” poniendo en la url un “action=alumnos”, de lo contrario, se muestra un mensaje de error de registro de alumno.

deleteStudentController: este controlador obtiene de la URL la matrícula del alumno a eliminar mediante “\$\_GET[“matricula”]” y se lo manda a un modelo llamado “Data::deleteStudentModel()” el cual tiene como parámetros la matrícula del alumno y el nombre de la tabla a usar “alumnos” y procede a hacer la eliminación del alumno, si la respuesta fue exitosa entonces re-direcciona a la vista “alumnos” poniendo en la URL un “action=alumnos” si no, entonces muestra un mensaje de error.

```

767 }
768 //Metodo que muestra a los alumnos de cada grupo que hayan realizado alguna sesion de cai
769 public static function teacherStudentsSessionController(){
770     $data = $_GET["id_grupo"];
771     $answer = Data::teacherStudentSessionsModel($data,"alumnos","horas_alumno");
772     foreach ($answer as $row => $student) {
773         echo'
774             <input type="hidden" id="inf" value="3'.$student["matricula"].'">
775             <tr>
776                 <td>'.$student["matricula"].'</td>
777                 <td>'.$student["nombre_alumno"].' '.$student["apellidos_alumno"].'</td>
778                 <td>'.$student["carrera"].'</td>
779                 <td>'.$student["email"].'</td>
780                 <td>'.$student["total_horas"].'</td><
781                 <td><a class="btn-floating btn-large waves-effect waves-light teal darken-3 btn modal-trigger" href="index.php?action=detallesSesion&matricula=
782                     '.$student["matricula"].'">i class="material-icons">assignment</i></a></td>
783             </tr>;
784         }
785         echo"</tbody>
786         </table>";
787     }
788 //Metodo que muestra los detalles de la sesion que realizo un alumno
789 public static function studentHoursDetailController(){
790     $data = $_GET["matricula"];
791     $answer = Data::studentHoursDetailModel($data,"horas_alumno","horas");
792     foreach ($answer as $row => $student) {
793         echo'
794             <input type="hidden" id="inf" value="3'.$student["matricula_alumno"].'">
795             <tr>
796                 <td>'.$student["actividad"].'</td>
797                 <td>'.$student["unidad"].'</td>
798                 <td>'.$student["fecha"].'</td>
799                 <td>'.$student["hora_entrada"].'</td>
800                 <td>'.$student["hora_salida"].'</td>
801             </tr>;
802         }
803         echo"</tbody>
804         </table>";
805     }
806 }
807

```

teacherStudentsSessionController: este controlador primeramente obtiene el id del grupo usando “\$\_GET[“id\_grupo”]” de la URL y la almacena en una variable “\$data” después procede a utilizar un modelo llamado “Data::teacherStudentsSessionModel()” el cual recibe como parámetros el id del grupo obtenido anteriormente almacenado en la variable “\$data”, el nombre de la tabla “alumnos” y el nombre de la tabla “horas\_alumno” y este modelo traerá los datos de ese alumno, y se almacenarán en la variable “\$answer” después se imprime en un ciclo “foreach” los datos del array “\$answer” imprimiéndolos como tabla, el nombre del alumno, matrícula, carrera, email y todos sus datos agregando un botón para ir a ver los detalles de la sesión de ese alumno.

studentHoursDetailController: este controlador obtiene la matrícula del alumno desde la URL mediante “\$\_GET[“matricula”]” y la almacena en una variable “\$data” para después hacer una consulta con el modelo “Data::studentHoursDetailModel()” el cual recibe como parámetros la matrícula del alumno, el nombre de la tabla “horas\_alumno” y el nombre de la tabla “horas”, y este modelo nos devuelve los datos de ese alumno con sus horas, y se almacenan los resultados en un array “\$answer” y después se imprimen en un ciclo mostrando los datos del alumno como su matrícula, actividades realizadas, la unidad, fechas, horas de entrada y horas de salida de ese alumno,

```

295 public static function showTeachersGroupsController(){
296     $data = $_SESSION["user"];
297     $answer = Data::showTeachersGroupsModel($data, "grupos");
298     foreach ($answer as $row => $group) {
299         echo '
300             <input type="hidden" id="inf" value="' . $group["id_grupo"] . '">
301             <tr>
302                 <td>' . $group["id_grupo"] . '</td>
303                 <td>' . $group["nombre_grupo"] . '</td>
304                 <td>' . $group["nivel"] . '</td>
305                 <td><a class="btn-floating btn-large waves-effect waves-light teal darken-3 btn modal-trigger" href="index.php?action=sessionAlumnos&id_grupo=' .
306                     $group["id_grupo"] . '"><i class="material-icons">assignment</i></a></td>
307             </tr>';
308     }
309     echo '</tbody>';
310     echo '</table>';
311 }
312
313 //Metodo para hacer la insercion de un grupo en la base de datos, utilizando el modelo de addGroupModel, para realizar la consulta en la base de datos e
314 //insertar un nuevo grupo
315 public static function addGroupController(){
316     if(isset($_POST["add"])){
317         $data = array("nombre_grupo" => $_POST["nombre_grupo"], "nivel" => $_POST["nivel"], "teacher" => $_POST["teacher"]);
318         $answer = Data::addGroupModel($data, "grupos");
319         var_dump($data);
320         if($answer == "success"){
321             echo "<script>
322                 window.location='index.php?action=grupos';
323             </script>";
324         }else{
325             echo "Error al registrar";
326         }
327     }
328 }
329
330 //Metodo del controlador para tomar el id del grupo y mandarlo al modelo para realizar la eliminacion en la base de datos
331 public static function deleteGroupController(){
332     $data = $_GET["id_grupo"];
333     $answer = Data::deleteGroupModel($data, "grupos");
334 }

```

showTeachersGroupController: este controlador muestra el grupo y sus datos de los teachers, en específico del teacher en sesión. Primeramente trae el id del teacher en sesión mediante la variable “\$\_SESSION[“user”]” y este id se almacena en la variable “\$data” y después se usa el modelo “Data::showTeachersGroupController()” el cual recibe dos parámetros, el id del teacher y el nombre de la tabla de la base de datos “grupos”. Posteriormente se imprimen sus datos en un ciclo mostrando el id del grupo, el nombre del grupo, el nivel y su asignación.

addGroupController: este controlador agrega un nuevo grupo a la base de datos, verificando si “add” de grupos fue lanzado, entonces empieza a obtener los datos registrados del nuevo grupo en el formulario mediante “\$\_POST[]” todos los campos, para almacenarlos en una variable “\$answer”, después se usa el modelo “Data::addGroupModel()” el cual recibe como parámetros el array con todos los datos del nuevo grupo y el nombre de la tabla a operar “grupos”, si la inserción fue exitosa entonces re-direcciona a la vista “grupos” poniendo en la URL un “action=grupos”, de lo contrario mostrará un mensaje de error en la agregación del nuevo grupo.

```

529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569

```

```

</optgroup>
<optgroup label="All Groups">
    MvcController::selectGroupController();
    echo'</optgroup>
</select>
</div>
<div class="input-field col s6">
    <input type="email" name="email" value="'.$answer["email"].'">
    <label for="email">E-mail</label>
</div>
</div>
<div class="modal-footer">
    <button type="submit" class="waves-effect waves-light blue darken-4 btn-small" name="update">Update Information</button>
</div>
</div>
</form>
';
}

//Este metodo utilizara los datos que se encuentren dentro del metodo editGroupController, ya que son lo que se utilizaran para hacer la modificacion del grupo
public static function updateStudentController(){
    if(isset($_POST["update"])){
        $data = array("matricula"=>$_POST["matricula"],"nombre_alumno"=>$_POST["nombre_alumno"],"apellidos_alumno"=>$_POST["apellidos_alumno"],"carrera"=>$_
        POST["carrera"],"grupo"=>$_POST["grupo"],"email"=>$_POST["email"]);
        $answer = Data::updateStudentModel($data, "alumnos");
        if($answer == "success"){
            echo "<script>
            window.location='index.php?action=alumnos';
            </script>";
        }
        else{
            echo "Error al Actualizar";
        }
    }
}

=====
===== HOURS CONTROLLER =====

```

updateStudentController: este controlador realiza la actualización de los datos de un estudiante en la base de datos, primeramente verificando si el botón “update” fue lanzado, entonces procede a obtener todos los datos del formulario llenado sobre un alumno específico, obteniéndolos mediante “\$\_POST[]” lo que sea que tengan los campos lo guardamos en un array y este array mas el nombre de la tabla “alumnos”, se lo mandamos a un modelo llamado “Data::updateStudentModel()” el cual realiza la función de actualizar el registro de ese alumno, si la actualización de datos en la base de datos fue exitosa entonces re-direcciona a la vista alumnos poniendo “action=alumnos”, de lo contrario mostrará un mensaje de error de actualización del alumno.

```

602     }
603     }
604     }
605     }
606     }
607     }
608     //Metodo que muestra todas las sesiones de cai que estan activas
609     public static function showHourController(){
610         if(isset($_SESSION["horas"])){
611             foreach($_SESSION["horas"] as $row => $matriculas) {
612                 echo'
613
614                 <tr>
615                     <td>' . $matriculas["matricula"] . '</td>
616                     <td>' . $matriculas["nombre"] . '</td>
617                     <td>' . $matriculas["actividad"] . '</td>
618                     <td>' . $matriculas["hora_entrada"] . '</td>
619                     <td>' . $matriculas["fecha"] . '</td>
620                 </td>
621
622                 <form method="post">
623                     <input type="hidden" name="matricula" value="' . $matriculas["matricula"] . '">
624                     <input type="hidden" name="nombre" value="' . $matriculas["nombre"] . '">
625                     <input type="hidden" name="actividad" value="' . $matriculas["actividad"] . '">
626                     <input type="hidden" name="hora_entrada" value="' . $matriculas["hora_entrada"] . '">
627                     <input type="hidden" name="fecha" value="' . $matriculas["fecha"] . '">
628                     <button type="submit" name="finish" class="waves-effect waves-light red darken-4 btn-large">Finish CAI hour</button>
629                 </form>
630                 <td>
631                     <form method="post">
632                         <button type="submit" name="finish" class="waves-effect waves-light yellow darken-4 btn-large">Cancel CAI hour</button>
633                     </form>
634                 </td>
635             </tr>
636         ' . '
637     ' . '
638     }
639     }
640
641     public static function finishHourController(){
642         $tiempo = 3000;
643         $tiempoMaximo = 3600;

```

showHourController: este controlador obtiene las horas mediante “\$\_SESSION[“horas”]” para después ir mostrando esos datos esos valores almacenados en la variable “\$\_SESSION[“horas”]” e ir formando una tabla una fila cada vuelta del ciclo “foreach” mostrando los datos como matrícula, nombre, actividad, hora de entrada de los alumnos que han realizado horas de cai en la hora, dando finalmente la opción de finalizar la hora de CAI, pero para realizar esto se deben cumplir algunas otras restricciones de horarios.

```

263 //Metodo que mostrara todos los grupos que se encuentran dentro de la base de datos, mediante una table
264 public static function showGroupsController(){
265     //por cada registro de grupos que se encuentre en la base de datos, se imprimiran dichos datos
266     $answer = Data::showGroupsModel("grupos","teachers");
267     foreach ($answer as $row => $group) {
268         echo'
269         <tr>
270             <td>'. $group['id_grupo']. '</td>
271             <td>'. $group['nombre_grupo']. '</td>
272             <td>'. $group['nivel']. '</td>
273             <td>'. $group['nombre_teacher']. '</td>
274             <td><a class="btn-floating btn-large waves-effect waves-light teal darken-3 btn modal-trigger" href="#"'. $group["id_grupo"]. "'><i
275             class="material-icons">edit</i></a></td>
276             <td><a class="btn-floating btn-large waves-effect waves-light red darken-4 btn modal-trigger" href="#delete'. $group["id_grupo"]. "'><i
277             class="material-icons">delete</i></a></td>
278         </tr>
279
280         <div id="delete'. $group["id_grupo"]. "' class="modal modal-fixed-footer" style="height:270px">
281             <div class="modal-content">
282                 <h4>Are you sure you want to delete this teacher?</h4><br>
283                 <button class="waves-effect waves-light red darken-4 btn-large modal-close">No</button>
284                 <a class="waves-effect waves-light blue darken-4 btn-large" href="index.php?action=eliminarGrupo&id_grupo=''. $group["id_grupo"]. "'>Yes</a>
285             </div>
286         </div>
287         '
288     }
289     echo "</tbody>
290     </table>";
291     foreach ($answer as $row => $group) {
292         MvcController::editGroupController($group["id_grupo"]);
293     }
294 }
295
296 public static function showTeachersGroupsController(){
297     $data=$SESSION["user"];
298     $answer = Data::showTeachersGroupsModel($data,"grupos");
299     foreach ($answer as $row => $group) {
300         echo'
301         <input type="hidden" id="inf" value="3'. $group["id_grupo"]. "'>
302         <tr>
303             <td>'. $group['id_grupo']. '</td>

```

showGroupsController: este controlador hace una consulta en la base de datos, mediante el modelo "Data::showGroupsController" recibiendo como parámetros el nombre de la tabla "grupos" y el nombre de la tabla "teachers", para de esta manera almacenar en un array todos los resultados obtenidos por el modelo sobre los grupos y sus teachers. Después se imprime mediante un ciclo el array con los datos de los grupos, mostrando el nombre del grupo, id del grupo, nivel, el nombre del teacher, y también puede editar el grupo y eliminarlo mediante modales.

```

224
225
226
227
228
229
230 #####3
231 #####
232
233 //Metodo para poder visualizar el nombre de los grupos en un select y que en cada opcion el valor contenga el id del grupo
234 public static function selectTeacher(){
235     $answer = Data::showTeachersModel("teachers");
236     foreach ($answer as $row => $teacher) {
237         echo "<option value='{$teacher[num_employed]}'>{$teacher[nombre_teacher]}</option>";
238     }
239 }
240
241 //Metodo para poder visualizar el nombre de los grupos en un select y que en cada opcion el valor contenga el id del grupo
242 public static function selectGroupController(){
243     $answer = Data::selectGroupModel("grupos");
244     foreach ($answer as $row => $group) {
245         echo "<option value='{$group[id_grupo]}'>{$group[nombre_grupo]}</option>";
246     }
247 }
248
249 //Metodo para poder visualizar el nombre de las alumnas en un select y que en cada opcion el valor contenga el id de la alumna
250 public static function selectStudent(){
251     $answer = Data::showStudentsModel("alumnos","grupos");
252     foreach ($answer as $row => $student) {
253         echo "<option value='{$student[matricula]}'>{$student[matricula]}</option>";
254     }
255 }
256
257
258
259 #####
260 ##### GRUPOS CONTROLLER #####
261 #####
262
263 //Metodo que mostrara todos los grupos que se encuentran dentro de la base de datos, mediante una table
264 public static function showGroupsController(){
265     //por cada registro de grupos que se encuentre en la base de datos, se imprimiran dichos datos

```

selectTeacher: esta función nos permite mostrar una lista de todos los teacher mediante una consulta en la base de datos mediante el modelo “Data::showTeachersModel” e ir imprimiéndolos en un ciclo, de igual manera lo hace con los otros dos controladores: “selectGroupController” y “selectStudent”, realiza consultas a la base de datos en una tabla específica trayendo así, los registros de cada tabla y después son mostrados en pantalla mediante un ciclo el cual imprime el id o matrícula ya sea del grupo o alumno, y el nombre.

```

808 //Metodo que permite visualizar las unidades
809 public static function showUnitsController(){
810     $answer = Data::showUnitsModel("unidad");
811     foreach ($answer as $row => $unit) {
812         echo "<tr>
813             <td>{$unit[id_unidad]}</td>
814             <td>{$unit[fecha_inicio]}</td>
815             <td>{$unit[fecha_termino]}</td>
816             <td><a class='btn-floating btn-large waves-effect waves-light teal darken-3 btn-modal-trigger' href='#{$unit[id_unidad]}'><i
            class='material-icons'>edit</i></a></td>
817         </tr>";
818     }
819     echo"</tbody>";
820     echo"</table>";
821     foreach ($answer as $row => $unit) {
822         MvcController::editUnitController($unit[id_unidad]);
823     }
824 }
825
826
827 //Metodo que permite la edición de las unidades
828 public static function editUnitController($data){
829     $answer = Data::editUnitModel($data,"unidad");
830     echo'
831     <form method="post" enctype="multipart/form-data">
832     <div id='{$answer[id_unidad]}' class="modal modal-fixed-footer">
833     <div class="modal-content">
834     <h4>Info</h4>
835     <div class="row">
836     <div class="input-field col s12">
837     <input type="hidden" name="matricula" value='{$answer[id_unidad]}'>
838     </div>
839     <div class="input-field col s12">
840     <input type="date" name="fecha_inicio" value='{$answer[fecha_inicio]}'>
841     <label for="nombre_grupo">Start Date</label>
842     </div>
843     <div class="input-field col s12">
844     <input type="date" name="fecha_termino" value='{$answer[fecha_termino]}'>
845     <label for="apellidos_alumno">End Date</label>
846     </div>
847     </div>
848

```

showUnitsController: este controlador hace una consulta a la base de datos mediante el modelo “Data::showUnitsModel(” que recibe como parámetro el nombre de la tabla “unidad” y devuelve el resultado de esa tabla con sus registros y son impresos mediante un ciclo, el id de la unidad, fecha de inicio, de termino y editarlas.



```

676 //Metodo para cancelar la hora de cai
677 public static function cancelHourController(){
678     if(isset($_POST["cancelar"])){
679         foreach ($SESSION["horas"] as $row => $alumno) {
680             if($alumno["matricula"]==$_POST["matricula"]){
681                 session_unset($SESSION["horas"]);
682             }
683         }
684     }
685 }
686 //Metodo para terminar todas las sesiones de cai
687 public static function finishAllHoursController(){
688     if(isset($_POST["finishall"])){
689         if(date("i")>55||date("i")>50 && date("i")<=59){//se determinara si la hora es apropiada para terminar la hora de cai, si lo es se realizara lo
        siguiente
690             $unidad = MvcController::determineUnit();
691             foreach ($SESSION["horas"] as $row => $sesion) { //Por cada sesion de cai se realizara la asignacion de los valores de la sesion a un arreglo, el
        cual sera registrado como parametro para terminar cada una de las horas de cai.
692                 $arreglohora = array("hora_entrada"=>$SESSION["horas"][$row_entrada], "hora_salida"=>date("H:i:s"), "fecha"=>date("Y-M-D"), "matricula"=>$
        SESSION["horas"][$matricula], "actividad"=>$SESSION["horas"][$actividad], "unidad"=>$unidad);
693                 $answer = Data::finishHourModel($arreglohora, "horas", "horas_alumno");
694             }
695             session_unset($SESSION["horas"]); //Se terminaran las horas de cai
696         }else{//Si la hora no es adecuada para terminar, se determinara que no se puede acabar la sesio
697             echo"You can't end up all cai hours yet";
698         }
699     }
700 }
701 //Metodo para determinar la unidad de la sesion de cai que se esta realizando, dependiendo de las fechas en la cual se esta realizando se determinara para que
        unidad se esta realizando a hora de cai.
702 public static function determineUnit(){
703     $answer = Data::showUnitsModel("unidad");
704     foreach ($answer as $row => $unit) {
705         if($unit["id_unidad"]==1){
706             if(date("Y-m-d")>$unit["fecha_inicio"] && date("Y-m-d")<=$unit["fecha_termino"]){
707                 return 1;
708             }else{
709                 break;
710             }
711         }else if($unit["id_unidad"]==2){
712             if(date("Y-m-d")>$unit["fecha_inicio"] && date("Y-m-d")<=$unit["fecha_termino"]){

```

El método `cancelHourController()` se encarga de cancelar la hora que se seleccione, tomando la matricula del alumno seleccionado, se hace una comparación con todas las matriculas agregadas en el arreglo de sesión, si se encuentra una coincidencia se elimina la hora y no se registra en la base de datos.

El método `finishAllHouesController()` se encarga de terminar todas las sesiones de cai siempre y cuando se cumpla la restricción de la hora de termino. Se determina la unidad en la cual se va registrar la hora y después se hace un ciclo, en dicho ciclo mientras se encuentren registros en el arreglo de sesión, seguirá su curso, mientras el ciclo este en curso se asignara un arreglo con los valores de la sesión en dicha posición del arreglo, dichos datos se mandaran al modelo el cual finaliza la sesión de cai.

```

733 //Metodo para mostrar los teachers que tienen alumnos que realizaron sesion de cai.
734 public static function sessionsController(){
735     $answer = Data::showTeachersModel("teachers");
736     foreach ($answer as $row => $sesion) {
737         if($sesion["num_empleado"]==1){
738             echo'
739                 <tr>
740                     <td>$sesion["nombre_teacher"].'.'.'$sesion["apellidos_teacher"].'</td>
741                     <td><a class="btn btn-floating btn-large waves-effect waves-light teal darken-3 btn modal-trigger"
742                         href="index.php?action=sesionGrupos&num_empleado='.$sesion["num_empleado"].'">Check</a></td>
743                 </tr>
744             ';
745         }
746     }
747     echo "</tbody>";
748     echo "</table>";
749 }
750 //Metodo que Visualiza la sesion de los alumnos de cada grupo que tiene cada teacher
751 public static function teacherSessionsController(){
752     $data = $_GET["num_empleado"];
753     $answer = Data::showTeachersGroupsModel($data, "grupos");
754     foreach ($answer as $row => $group) {
755         echo'
756             <input type="hidden" id="inf" value="3' . $group["id_grupo"] . '">
757             <tr>
758                 <td>$group["nombre_grupo"].'</td>
759                 <td>$group["nivel"].'</td>
760                 <td><a class="btn btn-floating btn-large waves-effect waves-light teal darken-3 btn modal-trigger" href="index.php?action=sesionAlumnos&id_grupo=' .
761                     $group["id_grupo"] . '">i class="material-icons">assignment</i></a></td>
762             </tr>
763         ';
764     }
765     echo "</tbody>";
766     echo "</table>";
767 }
768 }

```

El método `determineUnit()` se encarga de determinar cuál es la unidad en la que se encuentra dependiendo de la fecha actual, ya que cada unidad cuenta con su fecha de inicio y fecha de término y la fecha actual debe encontrarse en ese rango de fechas. El metodo `sessionContoller()` es utilizado para mostrar la tabla de los teacher, agregando un boton

el cual dara paso a revisar los grupos de cada teacher y poder revisar los alumnos de cada grupo y sus sesiones de CAI.

teacherSessionController() muestra los grupos del teacher seleccionado con la opcion de ver los alumnos que se encuentran dentro de ese grupo.

```
641 //Metodo que termina con la sesion de cai de un alumno
642 public static function finishHourController(){
643     $tiempo = 3000;
644     $tiempoMaximo = 3600;
645     $var = 0; //Variable que determinara si se puede terminar la sesion o no
646     if(isset($_POST["finish"])){
647         foreach ($SESSION["horas"] as $row => $alumno) {
648             if($alumno["matricula"]==$_POST["matricula"]){
649                 $T = time()-$alumno["tiempo"];
650                 echo $T;
651                 if($T<$tiempo){ //Si el tiempo en segundos es menor a 3000 var=1
652                     $var=1;
653                 }else if($T>$tiempoMaximo){ // Si el tiempo pasa de los 3600 var = 2 y la sesion terminara
654                     $var=2;
655                     session_unset($SESSION["horas"]);
656                 }else if($T>=$tiempo && $T<=$tiempoMaximo){ // Si el tiempo es mayor o igual a 3000 y el tiempo en menor o igual a 3600, var=3 y terminara la
657                     session
658                     $var=3;
659                     session_unset($SESSION["horas"]);
660                 }
661             }
662         }
663     }
664     if($var==1){ //Si var es igual a 1, entonces la sesion no puede terminar
665         echo "The session can not be finish before a 50 minutes lapse";
666     }else if($var == 2){ //Si var es igual a 2, la sesion terminara por que se paso el tiempo de terminar la sesion
667         echo "The student exceed the limit to finish the session";
668     }else if($var==3){ // Si var es igual a 3, entonces se termino la sesion exitosamente y se realizara la insercion a la base de datos de dicha sesion de
669         cai
670         $unidad = MvcController::determineUnit();
671         echo "Session Concluded";
672         $arregloHora = array("hora_entrada"=>$_POST["hora_entrada"],"hora_salida"=>date("H:i:s"),"fecha"=>$_POST["fecha"],"matricula"=>$_POST["matricula"],
673             "actividad"=>$_POST["actividad"],"unidad"=>$unidad);
674         $answer = Data::finishHourModel($arregloHora,"horas","horas_alumno");
675     }
```

El metodo finishHourController() es el metodo encargado de terminar las sesiones de CAI cuando esten activas, determinando el tiempo minimo que se tiene que estar dentro de la sesion, dicho tiempo se le asigna a una variable \$tiempo = 3000, que en minutos es la representacion a 50 minutos, y un \$tiempoMaximo = 3600 que es equivalente a 60 minutos. Mediante una variable que va controlando la decision que se tomara se le asignara un numero a \$var, se condiciona dependiendo del tiempo que este transcurriendo, si el tiempo que el alumno tiene en el aula(\$alumno["tiempo"]) es menor al \$tiempo entonces no se puede terminar la hora, si el tiempo del alumno es mayor al \$tiempoMaximo, se determina que tampoco puede dar fin a la sesion. Pero si el tiempo del alumno esta entre el rango del \$tiempo y \$tiempoMaximo, la sesion puede concluir exitosamente.

```

568 ===== HOURS CONTROLLER =====
569
570 //Metodo para agregar una hora de cai
571 public static function addHourController(){
572     if(isset($_POST["Iniciar"])){//Si se hace uso del metodo post con el boton de nombre iniciar se hara lo siguiente
573
574         $numero = 0; //Variable que determinara si un alumno se encuentra dentro de cai o no
575         $nombre = Data::editStudentModel($_POST["matricula"],"alumnos","grupos");//Se consultan los datos del alumno dependiendo de la matricula
576         if($nombre){ //Se condiciona si si existe dicho alumno
577
578             //Se declara un arreglo con todos los datos que se necesitan para registrar la hora
579             $arreglo = array("matricula"=>$_POST["matricula"],"actividad" =>$_POST["actividad"],"nombre"=>$nombre["nombre_alumno"]." ".$nombre["
580             apellidos_alumno"],"hora_entrada"=>date("H:i"),"fecha"=>date("Y-m-d"),"tiempo"=>time());
581             //Por cada sesion que se encuentra dentro del arreglo de sesion en hora, se hara una comparacion para ver si el alumno ya esta dentro de cai o no
582             foreach ($SESSION["horas"] as $row => $matriculas) {
583                 if($matriculas["matricula"]==$arreglo["matricula"]){
584                     $numero++;
585                     break;
586                 }
587             }
588             //Se condiciona si el tiempo en minutos es menor a 5, entonces el alumno puede ingresar a cai
589             if(date("i")<05){
590                 if($numero==1){
591                     echo "This student is already in CAI";
592                 }else{
593                     $SESSION["horas"][] = $arreglo;
594                 }
595             }else{
596                 //Si se determina que se excedio el tiempo de registro de sesion en cai
597                 echo "Exceed Limit time to add a student to cai";
598             }
599         }else{
600             echo "This student isn't registered in the system";//Si la consulta no fue exitosa se determinara que el alumno no esta registrado
601         }
602     }
603 }
604 }
605 }
606 }
607 }

```

El método `addHourController()` se encarga de registrar las horas de CAI, este método se inicia si se utilizó el método POST en el elemento de nombre "Iniciar". Se declara una variable `$numero=0`, esta variable será la que determinara si el alumno que se registrara ya se encuentra realizando una sesión o no. La variable `$nombre` será igual a los datos del alumno del cual se ingresó la matricula. Si dicho registro existe en la base de datos, se creara un arreglo el cual contendrá todos los datos que se necesitan para registrar un sesión, como lo es la matricula del alumno, hora de inicio, hora de salida, fecha, actividad. Después se determina si La hora de entrada del alumno es válida, ya que el sistema permite entrar hasta que el marcador numérico de minutos sea menor o igual a 05, si este es mayor no se permitirá la entrada al alumno.

## MODELO

```

1 <?php
2 //Clase que es llamada en los modelos para poder realizar las consultas con la
  base de datos
3 class Connection{
4     public static function connect(){
5         $db = new PDO("mysql:host=localhost;dbname=cai","root","");//Variable $db
        contendra el valor de la conexion a la base de datos mediante pdo
6         return $db;//Se devolvera dicho dicho valor
7     }
8 }
9
10 ?>
11

```

Este Fragmento de código es el encargado de hacer la conexión de la base de datos. Primero creando una clase llamada `Connection`, la cual contiene el método que realiza la conexión con la base de datos. El método `connect()` el cual realiza la conexión, cuenta con una variable la cual contendrá dicha conexión. Se declara como un nuevo objeto de PDO con los siguientes parámetros, el host,

nombre de la base de datos, usuario y contraseña, los cuales depende del servidor de base de datos que se este utilizando y las credenciales que tenga dicha base de datos.

```
1 <?php
2 require_once "connection.php";
3 class Data extends Connection{
4
5     //Metodo para mostrar la informacion de todas las alumnas de la academia
6     public static function showStudentsModel($table,$table2){
7         $statement = Connection::connect()->prepare("SELECT a.nombre_alumno, a.apellidos_alumno, a.matricula, a.carrera, a.email, g.nombre_grupo, g.nivel, t.
8             nombre_teacher, t.apellidos_teacher FROM $table as a INNER JOIN $table2 as g on a.grupo = g.id_grupo INNER JOIN teachers as t on g.teacher = t.
9             num_empleado");
10
11         $statement->execute();
12         return $statement->fetchAll();
13     }
14
15     //Metodo para mostrar todos los estudiantes dependiendo del grupo
16     public static function showStudentsGroupModel($data,$table){
17         $statement = Connection::connect()->prepare("SELECT * FROM $table WHERE grupo = :grupo");
18         $statement->bindParam(":grupo",$data,PDO::PARAM_INT);
19         $statement->execute();
20         return $statement->fetchAll();
21     }
22
23
24     //Metodo para realizar la consulta en la base de datos y poder realizar la insercion en la tabla alumnas, los datos a insertar seran el arreglo que se
25     recibira como parametro.
26     public static function addStudentModel($data,$table){
27         $statement = Connection::connect()->prepare("INSERT INTO $table(matricula,nombre_alumno,apellidos_alumno,carrera,grupo,email) VALUES
28             (:matricula,:nombre_alumno,:apellidos_alumno,:carrera,:grupo,:email)");
29         $statement->bindParam(":matricula",$data["matricula"],PDO::PARAM_INT);
30         $statement->bindParam(":nombre_alumno",$data["nombre_alumno"],PDO::PARAM_STR);
31         $statement->bindParam(":apellidos_alumno",$data["apellidos_alumno"],PDO::PARAM_STR);
32         $statement->bindParam(":carrera",$data["carrera"],PDO::PARAM_STR);
33         $statement->bindParam(":grupo",$data["grupo"],PDO::PARAM_INT);
34         $statement->bindParam(":email",$data["email"],PDO::PARAM_STR);
35         if($statement->execute()){
36             return "success";
37         }else{
38             return "fail";
39         }
40     }
41 }
```

Se utiliza el metodo de PHP `require_once`, para hacer el llamado hacia el archivo `connection.php` el cual contiene la conexión hacia la base de datos y de esta manera poder hacer uso de la clase `Connection` y asi poder crear la clase `Data` que herede de la clase `Connection`.

El metodo `showStudentModel()` recibe como parametro de entrada dos variables, `$table` y `$table2`, las cuales son el nombre de las tablas "alumnos" y "grupos". La variable `$statement` se crea como un objeto de la clase `Connection`, otorgando el metodo `connection()` dando paso a que se realice la consulta mediante la sentencia `prepare()`, dentro de esta se encuentra la sentencia `SELECT` de SQL la cual realizara la extraccion de los datos haciendo inner join entre las tablas "alumnos" y "grupos" para asi poder mostrar de manera correcta los datos de cada alumno.

Despues se encuentra el metodo `showStudentsGroupModel()`, el cual lo que hace es mostrar los alumnos que se encuentra en cada grupo dependiendo del identificador del grupo, dicho metodo recibe como parametro la variables `$data` y `$table`, las cuales son "id del grupo" y la tabla "alumnos", para asi mediante el id de grupo desplegar todos los alumnos que contengan ese id en sus registros.

El metodo `addStudentController()`, es el metodo encargado de hacer la insercion hacia la base de datos, como en los demas metodos tambien se reciben parametros de entrada, los cuales seran la variables `$data` y `$table`, las cuales contendran un arreglo con todo la infomacion que se registro del alumno y la tabla "alumnos". La variable `$statement` contendra un consulta `INSERT` de SQL la cual hara la insercion en la base de datos para registrar un nuevo alumno, obteniendo como informacion del alumno su matricula, su nombre, apellidos, su carrera, su grupo de ingles y el email. Esta informacion es pasada mediante un metodo de seguridad de datos de PDO el cual es `bindParam`, que mantiene la integridad de los datos, previniendolos de inyecciones de SQL.

```

40 //Metodo para realizar la consulta que devolviera los datos de la alumna la cual se desea editar
41 public static function editStudentModel($data,$table,$table2){
42     $statement = Connection::connect()->prepare("SELECT * FROM $table as a INNER JOIN $table2 as g on a.grupo = g.id_grupo WHERE matricula = :matricula");
43     $statement->bindParam(":matricula",$data,PDO::PARAM_INT);
44     $statement->execute();
45     return $statement->fetch();
46 }
47
48 //Metodo para editar los datos de la alumna dependiendo de los datos que se obtuvieron en la funcion editStudentModel.
49 public static function updateStudentModel($data,$table){
50     $statement = Connection::connect()->prepare("UPDATE $table SET nombre_alumno = :nombre_alumno, apellidos_alumno=:apellidos_alumno, carrera = :carrera, grupo = :grupo, email=:email WHERE matricula = :matricula");
51     $statement->bindParam(":nombre_alumno",$data["nombre_alumno"],PDO::PARAM_STR);
52     $statement->bindParam(":apellidos_alumno",$data["apellidos_alumno"],PDO::PARAM_STR);
53     $statement->bindParam(":carrera",$data["carrera"],PDO::PARAM_STR);
54     $statement->bindParam(":email",$data["email"],PDO::PARAM_STR);
55     $statement->bindParam(":grupo",$data["grupo"],PDO::PARAM_INT);
56     $statement->bindParam(":matricula",$data["matricula"],PDO::PARAM_INT);
57     if($statement->execute()){
58         return "success";
59     }else{
60         return "fail";
61     }
62 }
63
64 //Metodo que realizara la eliminacion de la alumna dependiendo del id que se haya recibido como parametro
65 public static function deleteStudentModel($data,$table){
66     $statement = Connection::connect()->prepare("DELETE FROM $table WHERE matricula = :matricula");
67     $statement->bindParam(":matricula",$data,PDO::PARAM_INT);
68     if($statement->execute()){
69         return "success";
70     }else{
71         return "fail";
72     }
73 }

```

El metodo `editStudentModel()` recibe como parametro las variable `$data`, `$table` y `$table2`, dichas variables serian la matricula del alumno, la tabla alumnos y la tabla grupos, respectivamente, se realiza una sentencia `SELEEC` de SQL para obtener los datos del alumno a editar dependiendo de la matricula.

El metodo `updateStudentModel()` recibe como parametro la variable `$data` y `$table`, que son un arreglo que contiene todos los datos que se ingresan en la consulta y la tabla alumnos que es la cual se le hace la actualizacion. Se realiza una sentencia `UPDATE` de SQL y se llena con los datos ingresado y almacenados en el arreglo.

El metodo `deleteStudentModel()` recibe como parametro las variables `$data` y `$table`, las cuales son la matricula del alumno y la tabla alumnos respectivamente. Mediante el uso de la matricula como referencia para realizar la eliminacion del alumno con la sentencia `DELETE` de SQL.

```

73     }
74
75     //Metodo que muestra todos los grupos, para ser vistos por medio de un select
76     public static function selectGroupModel($stable){
77         $statement = Connection::connect()->prepare("SELECT * FROM $stable");
78         $statement->execute();
79         return $statement->fetchAll();
80         $statement->close();
81     }
82
83
84     //Metodo para mostrar la informacion de todos los grupos de la academia
85     public static function showGroupsModel($stable,$stable2){
86         $statement = Connection::connect()->prepare("SELECT * FROM $stable as g inner join $stable2 as t WHERE g.teacher = t.num_empleado");
87         $statement->execute();
88         return $statement->fetchAll();
89         $statement->close();
90     }
91
92     //Metodo que realiza la consulta que contendra todos los grupos de un teacher seleccionado
93     public static function showTeachersGroupsModel($data,$stable){
94         $statement = Connection::connect()->prepare("SELECT * FROM $stable WHERE teacher =:teacher");
95         $statement->bindParam("teacher",$data,PDO::PARAM_INT);
96         $statement->execute();
97         return $statement->fetchAll();
98         $statement->close();
99     }
100
101     //Metodo para realizar la insercion de un grupo en la base de datos,.
102     public static function addGroupModel($data,$stable){
103         $statement = Connection::connect()->prepare("INSERT INTO $stable(nombre_grupo,nivel,teacher) VALUES (:nombre_grupo,:nivel,:teacher)");
104         $statement->bindParam(":nombre_grupo",$data["nombre_grupo"],PDO::PARAM_STR);
105         $statement->bindParam(":nivel",$data["nivel"],PDO::PARAM_INT);
106         $statement->bindParam(":teacher",$data["teacher"],PDO::PARAM_INT);
107
108         if($statement->execute()){
109             return "success";
110         }else{
111             return "fail";
112         }
113     }

```

El metodo selectGroupModel() solamente recibe como parametro la variable \$stable la cual representa la tabla “grupos”. Este metodo realiza mediante una consulta SELECT de SQL, la recoleccion de todos los datos de todos los registros de dicha tabla, despues los manda a un arreglo y los retorna para asi poder utilizar dicho arreglo en el controlador.

El metodo showGroupsModel() recibe como parametros dos variables, \$stable y \$stable2, las cuales son la tabla “grupos” y la tabla “teachers”, ya que es necesario utilizar la tabla teachers para poder visualizar el nombre del teacher al cual esta relacionado el grupo. Mediante una sentencia SELECT de SQL se realiza la extraccion de los datos, haciendo un INNER JOIN dando como comparacion el numero de empleado del teacher y asi mostrar cual es el que corresponde a cada grupo.

El metodo showTeachersGroupModel() se encarga de hacer lo mismo que el metodo showGroupsModel, solamente que este muestra los grupos que tiene un teacher, mediante el numero de empleado, revisando cuanles grupos cuentan como referencia con ese numero.

El metodo addGroupModel() se encarga de recibir los datos de que se necesitan para realizar el registro de un grupo, los datos se reciben como un arreglo, dicho arreglo lo representa la variable \$data, y se recibe la tabla a la cual se asignara con la variable \$stable. Mediante bindParam se hace el paso de variables para que sea una manera mas segura y se tenga una buena integridad de datos.

```

115 //Metodo para realizar la consulta que devolvera los datos del grupo la cual se desea editar
116 public static function editGroupModel($data,$table,$table2){
117     $statement = Connection::connect()->prepare("SELECT * FROM $table as g inner join $table2 as t WHERE t.num_empleado = g.teacher AND g.id_grupo = :id_grupo");
118     $statement->bindParam(":id_grupo",$data,PDO::PARAM_INT);
119     $statement->execute();
120     return $statement->fetch();
121     $statement->close();
122 }
123
124 //Metodo para editar los datos del grupo dependiendo de los datos que se obtuvieron en la funcion editStudentModel.
125 public static function updateGroupModel($data,$table){
126     $statement = Connection::connect()->prepare("UPDATE $table SET nombre_grupo = :nombre_grupo,teacher=:teacher WHERE id_grupo = :id_grupo");
127     $statement->bindParam(":nombre_grupo",$data["nombre_grupo"],PDO::PARAM_STR);
128     $statement->bindParam(":teacher",$data["teacher"],PDO::PARAM_INT);
129     $statement->bindParam(":id_grupo",$data["id_grupo"],PDO::PARAM_INT);
130
131     if($statement->execute()){
132         return "success";
133     }else{
134         return "fail";
135     }
136 }
137
138 //Metodo que realizara la eliminacion del grupo dependiendo del id que se haya recibido como parametro
139 public static function deleteGroupModel($data,$table){
140
141     $statement = Connection::connect()->prepare("DELETE FROM $table WHERE id_grupo = :id_grupo");
142     $statement->bindParam(":id_grupo",$data,PDO::PARAM_INT);
143     $statement->execute();
144     return "success";
145 }
146

```

El metodo editGroupModel, recibe como parametro las variables \$data, \$table y \$table2, la variable \$data representa el id del grupo para saber cual sera el grupo del cual se mostrara informacion, la variable \$table hace referencia a la tabla grupos y la variable \$table2 hace referencia a la table teachers, para asi poder mostrar los datos del teacher a cargo del grupo.

El metodo updateGroupModel() se encarga de actualizar la informacion que es recibida mediante el controlador, esta informacion entra como parametro de la funcion como un arreglo el cual se referencia como \$data, y la tabla que sera la tabla grupos. Se realiza la actualizacion de los datos mediante una sentencia UPDATE de SQL.

El metodo deleteGroupModel() se encarga de eliminar la informacion que es recibida mediante el controlador, esta informacion entra como parametro de la funcion como una variable la cual se referencia como \$data, y la tabla que sera la tabla grupos. Se realiza la actualizacion de los datos mediante una sentencia DELETE de SQL.

```

149 public static function loginModel($data,$table){
150     $statement = Connection::connect()->prepare("SELECT * FROM $table WHERE email = :email and password = :password");
151     $statement->bindParam(":email",$data["email"],PDO::PARAM_STR);
152     $statement->bindParam(":password",$data["password"],PDO::PARAM_STR);
153     $statement->execute();
154     return $statement->fetch();
155     $statement->close();
156 }
157
158 #####
159 ##### TEACHERS #####
160 #####
161
162 //Metodo que contendra toda la informacion de todos los teachers
163 public static function showTeachersModel($table){
164     $statement = Connection::connect()->prepare("SELECT * from $table");
165     $statement->execute();
166     return $statement->fetchAll();
167     $statement->close();
168 }
169
170 //Metodo que realizara el agregado hacia la base de datos de un teacher, recibiendo todos los datos requeridos para realizar las insercion en la base de
171 //datos, mediante el uso de bindParam, para hacer que los datos sean mas seguros.
172 public static function addTeacherModel($data,$table){
173     $statement = Connection::connect()->prepare("INSERT INTO $table(num_empleado,nombre_teacher,apellidos_teacher,email,password,foto,telefono) VALUES
174     (:num_empleado,:nombre_teacher,:apellidos_teacher,:email,:password,:foto,:telefono)");
175     $statement->bindParam(":num_empleado",$data["num_empleado"],PDO::PARAM_INT);
176     $statement->bindParam(":nombre_teacher",$data["nombre_teacher"],PDO::PARAM_STR);
177     $statement->bindParam(":apellidos_teacher",$data["apellidos_teacher"],PDO::PARAM_STR);
178     $statement->bindParam(":email",$data["email"],PDO::PARAM_STR);
179     $statement->bindParam(":password",$data["password"],PDO::PARAM_STR);
180     $statement->bindParam(":foto",$data["foto"],PDO::PARAM_STR);
181     $statement->bindParam(":telefono",$data["telefono"],PDO::PARAM_STR);
182     if($statement->execute()){
183         return "success";
184     }else{
185         return "fail";
186     }
187 }
188

```

El metodo loginModel() se encarga de verificar la informacion que es ingresada para iniciar la sesion en el sistema, se verifica el email y el password del usuario, si dicho password y dicho email coinciden en un mismo registro en la base de datos, se permitira el acceso hacia el sistema.

El metodo showTeachersModel() recibe como parametro la variable, \$table, la cual es la tabla "teachers", Mediante una sentencia SELECT de SQL se realiza la extraccion de los datos.

El metodo addTeacherModel() se encarga de recibir los datos de que se necesitan para realizar el registro de un teacher, los datos se reciben como un arreglo, dicho arreglo lo representa la variable \$data, y se recibe la tabla a la cual se asignara con la variable \$table. Mediante bindParam se hace el paso de variables para que sea una manera mas segura y se tenga una buena integridad de datos.

```
186
187 //Metodo para realizar el borrado de la informacion de un teacher en la base de datos, dependiendo de su identificador el cual es su numero de empleado
188 public static function deleteTeacherModel($data,$table){
189     $statement = Connection::connect()->prepare("DELETE FROM $table WHERE num_empleado = :num_empleado");
190     $statement->bindParam(":num_empleado",$data,PDO::PARAM_INT);
191     if($statement->execute()){
192         return "success";
193     }else{
194         return "fail";
195     }
196 }
197
198 //Metodo que realizara la edicion de la informacion de un teacher dependiendo de su numero de empleado, el cual es su identificador en la base de datos
199 public static function editTeacherModel($data,$table){
200     $statement = Connection::connect()->prepare("SELECT * FROM $table WHERE num_empleado = :num_empleado");
201     $statement->bindParam(":num_empleado",$data,PDO::PARAM_INT);
202     $statement->execute();
203     return $statement->fetch();
204     $statement->close();
205 }
206
207 //Metodo que realizara la actualizacion de un teacher tomando los valores correspondientes a cada parte de los campos en la base de datos.
208 public static function updateTeacherModel($data,$table){
209     $statement = Connection::connect()->prepare("UPDATE $table SET nombre_teacher = :nombre_teacher, apellidos_teacher = :apellidos_teacher, email = :email,
210     password = :password, foto = :foto, telefono = :telefono WHERE num_empleado = :num_empleado");
211     $statement->bindParam(":nombre_teacher",$data["nombre_teacher"],PDO::PARAM_STR);
212     $statement->bindParam(":apellidos_teacher",$data["apellidos_teacher"],PDO::PARAM_STR);
213     $statement->bindParam(":email",$data["email"],PDO::PARAM_STR);
214     $statement->bindParam(":password",$data["password"],PDO::PARAM_STR);
215     $statement->bindParam(":foto",$data["foto"],PDO::PARAM_STR);
216     $statement->bindParam(":telefono",$data["telefono"],PDO::PARAM_STR);
217     $statement->bindParam(":num_empleado",$data["num_empleado"],PDO::PARAM_INT);
218     if($statement->execute()){
219         return "success";
220     }else{
221         return "fail";
222     }
223 }
```

El metodo deleteTeacherModel() se encarga de eliminar la informacion que es recibida mediante el controlador, esta informacion entra como parametro de la funcion como una variable la cual se referencia como \$data, y la tabla que sera la tabla teachers. Se realiza la actualizacion de los datos mediante una sentencia DELETE de SQL.

El metodo editTeacherModel(), recibe como parametro las variables \$data, \$table, la variable \$data representa el numero de empleado para saber cual sera el teacher del cual se mostrara informacion, la variable \$table hace referencia a la tabla "teachers".

El metodo updateTeacherModel() se encarga de actualizar la informacion que es recibida mediante el controlador, esta informacion entra como parametro de la funcion como un arreglo el cual se referencia como \$data, y la tabla que sera la tabla "teachers". Se realiza la actualizacion de los datos mediante una sentencia UPDATE de SQL.



```

225 public static function finishHourModel($data,$table,$table2){
226     $statement = Connection::connect()->prepare("INSERT INTO $table(hora_entrada,hora_salida,fecha) VALUES (:hora_entrada,:hora_salida,:fecha)");
227     $statement->bindParam(":hora_entrada",$data["hora_entrada"],PDO::PARAM_STR);
228     $statement->bindParam(":hora_salida",$data["hora_salida"],PDO::PARAM_STR);
229     $statement->bindParam(":fecha",$data["fecha"],PDO::PARAM_STR);
230     $statement->execute();
231     $statement->fetch();
232
233     $statement2 = Connection::connect()->prepare("INSERT INTO $table2(id_hora,matricula,actividad,unidad) VALUES (:id_hora,:matricula,:actividad,:unidad)");
234     $statement2->bindParam(":id_hora",$data["id_hora"],PDO::PARAM_INT);
235     $statement2->bindParam(":matricula",$data["matricula"],PDO::PARAM_INT);
236     $statement2->bindParam(":actividad",$data["actividad"],PDO::PARAM_STR);
237     $statement2->bindParam(":unidad",$data["unidad"],PDO::PARAM_INT);
238     $statement2->execute();
239 }
240
241 //Metodo que muestra las unidades que existen
242 public static function showUnitsModel($table){
243     $statement = Connection::connect()->prepare("SELECT * FROM $table");
244     $statement->execute();
245     return $statement->fetchAll();
246     $statement->close();
247 }
248 //Metodo que permite editar la unidad de cai
249 public static function editUnitModel($data,$table){
250     $statement = Connection::connect()->prepare("SELECT * FROM $table WHERE id_unidad = :id_unidad");
251     $statement->bindParam(":id_unidad",$data,PDO::PARAM_INT);
252     $statement->execute();
253     return $statement->fetch();
254     $statement->close();
255 }
256
257 //Metodo que contiene el resultado de consultar las sesiones de los alumnos de cada teacher, dependiendo de su matricula y el id del grupo
258 public static function teacherStudentSessionsModel($data,$table,$table2){
259     $statement = Connection::connect()->prepare("SELECT *,COUNT(ha.matricula_alumno) as 'total_horas' FROM $table AS a INNER JOIN $table2 AS ha ON a.matricula
260     = ha.matricula_alumno WHERE grupo = :grupo");
261     $statement->bindParam(":grupo",$data,PDO::PARAM_INT);
262     $statement->execute();
263     return $statement->fetchAll();
264     $statement->close();
265 }
266 //Metodo que contendra la informacion de los detalles de cada sesion de cai de cada alumno
267 public static function studentHoursDetailModel($data,$table,$table2){
268     $statement = Connection::connect()->prepare("SELECT * FROM $table as ha INNER JOIN $table2 as h ON ha.id_hora =
269     h.id_hora WHERE matricula_alumno = :matricula_alumno");
270     $statement->bindParam(":matricula_alumno",$data,PDO::PARAM_INT);
271     $statement->execute();
272     return $statement->fetchAll();
273     $statement->close();
274 }
275
276 }
277
278 ?>

```

El metodo finishHourModel() recibe como parametros las variables \$data, \$table y \$table2, la variable \$data representa un arreglo con todos los datos que conforman el registro de una hora, matricula del alumno, hora de entrada,hora de salida, fecha, unidad y actividad, dichos datos son registrados correspondientemente en las tablas horas y hora\_alumno, la cuales contienen toda la informacion sobre la sesion de CAI que el alumno haya realizado. Mediante consultas INSERT INTO de SQL se realiza la insercion de los datos hacia las tablas.

El metodo showUnitModel() recibe como parametro la variable \$table la cual representa el nombre de la tabla "unidades", mediante una consulta SELECT de SQL, se hace la extraccion de los datos para poder mostrar toda la informacion de todas las unidades.

El metodo editUnitModel() recibe como parametro la variable \$data y la variable \$table, \$data es la variable que contiene el valor del identificador de la unidad ya que este es necesario para poder extraer la informacion de la unidad la cual se quiere editar, y la variable \$table se encarga de recibir el nombre de la tabla "unidades". Mediante una consulta SELECT de SQL y utilizando la funcion WHERE mandando como valor el identificador de la unidad, se retorna la informacion de la unidad deseada.

El metodo studentHourDetailModel() se encarga de mandar los detalles de cada una de las sesiones que ha realizado un estudiante.