



# UNIVERSIDAD DON BOSCO

**FACULTAD DE ESTUDIOS TECNOLOGICOS**

**ESCUELA DE COMPUTACIÓN**

**Catedrático:**

Delmy Azucena Majano Menjívar

**Integrantes:**

Pineda López, Ronald David PL171439

Méndez Arévalo, Julián Alejandro MA222711

Vargas Hernández, Kennet Adonay VH222730

Vásquez Rodríguez, Denis Josué VH222731

Velásquez Rodríguez, Andrés René VR222732

**Asignatura:**

Administración de proyectos ADP 404

**Fecha:**

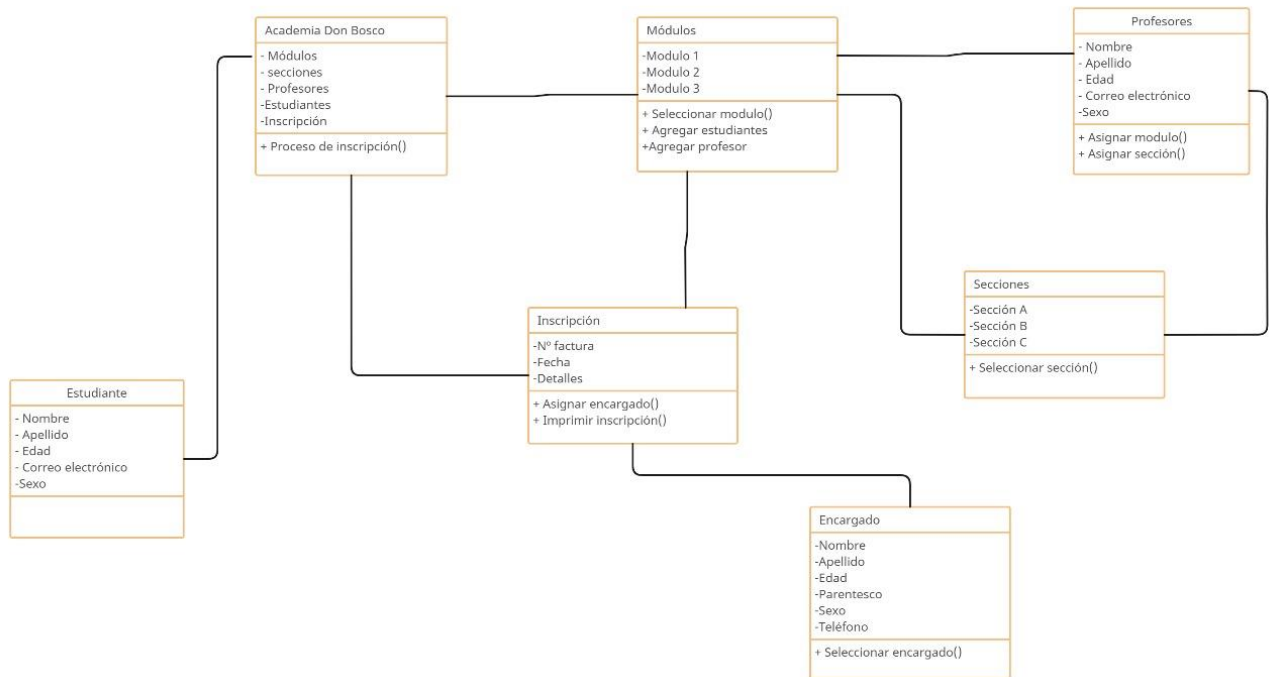
27/10/2023

# Índice

<b>1. Diseño de la capa de lógica de negocios .....</b>	<b>3</b>
<b>1.1. Diagrama de clases.....</b>	<b>3</b>
<b>1.2. Procedimientos.....</b>	<b>4</b>
<b>1.3. Funciones.....</b>	<b>5</b>
<b>1.4. Validaciones.....</b>	<b>6</b>
<b>2. Diseño de la interfaz de usuario.....</b>	<b>7</b>
<b>2.1. Diseño de pantallas .....</b>	<b>7</b>
<b>3. Elementos estáticos .....</b>	<b>12</b>
<b>4. Elementos dinámicos.....</b>	<b>13</b>
<b>5. Temas y estilos .....</b>	<b>17</b>
<b>6. Creación de las fuentes de datos y objetos .....</b>	<b>18</b>
<b>7. Programación de algunas clases, procedimientos funciones y validaciones.....</b>	<b>21</b>
<b>7.1. Procedimientos.....</b>	<b>21</b>
<b>7.2. Funciones.....</b>	<b>23</b>
<b>7.3. Validaciones.....</b>	<b>27</b>
<b>8. Creación de algunos formularios web.....</b>	<b>29</b>
<b>9. Actualización de cronogramas de actividades .....</b>	<b>31</b>













































































# 1. Diseño de la capa de lógica de negocios

## 1.1. Diagrama de clases



## 1.2.Procedimientos

La mayoría de los procedimientos dentro de la base de datos posee un menú, para fácil acceso a las funciones con el objetivo de simplificar el código dentro del propio aplicativo, además que se aumenta un poco más la seguridad en cuanto al acceso de consultas de la base de datos del aplicativo.

	Name	Type	Returns	
<input type="checkbox"/>	Last_modulo	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	Login_Personal	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	add_estudiante	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	add_modulo	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	add_seccion	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	buscar_estu	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	buscar_estu_seccion	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	buscar_modulo	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	delete_estu_seccion	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	delete_seccion	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	delete_seccion_estu	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	deletes	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	mostrar	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	mostrarSecciones	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	mostrar_alumnos	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	show_individual	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	update_estu	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	update_seccion	PROCEDURE		 Edit  Execute  Export  Drop
<input type="checkbox"/>	update_seccion_estu	PROCEDURE		 Edit  Execute  Export  Drop

### 1.3. Funciones

En cuanto a las funciones dentro del aplicativo son muchas, pero algunas de las más importantes tanto como para que este funciones correctamente y para generar algún tipo de validación extra.

Nombre de la función	Parámetros	Función
<b>AgregarEstudiante</b>	Ninguno	Función para agregar a un estudiante dentro del aplicativo, no necesita parámetros debido a que con JQuery se recogen los datos directamente de las entradas de texto del formulario, además que la respuesta al momento de agregar el estudiante es por medio de ajax, para demás formularios se ha realizado el mismo procedimiento
<b>Cerrar</b>	Ninguno	Función para eliminar cualquier modal que aparezca, se ejecuta al momento de hacer click en el botón "X" o "aceptar"
<b>mostrarEstudiante</b>	Nº de pagina	Función para mostrar todos los estudiantes por medio de Ajax, se pide numero de pagina para poder realizar la paginación, esta variable de forma inicial es 1, de igual manera se realiza para cualquier otra información que se quiera mostrar en otra parte del aplicativo.
<b>info</b>	Ninguno	Función para mostrar un modal para ver la información de un usuario en específico, el id del estudiante por ver se recoge por jquery a través de la propiedad "data_id".
<b>generarCodigo</b>	Apellido	Función de php para genera el código del estudiante dentro del sistema que será único, se genera a partir de los apellidos o apellido, agregando tres números aleatorios.

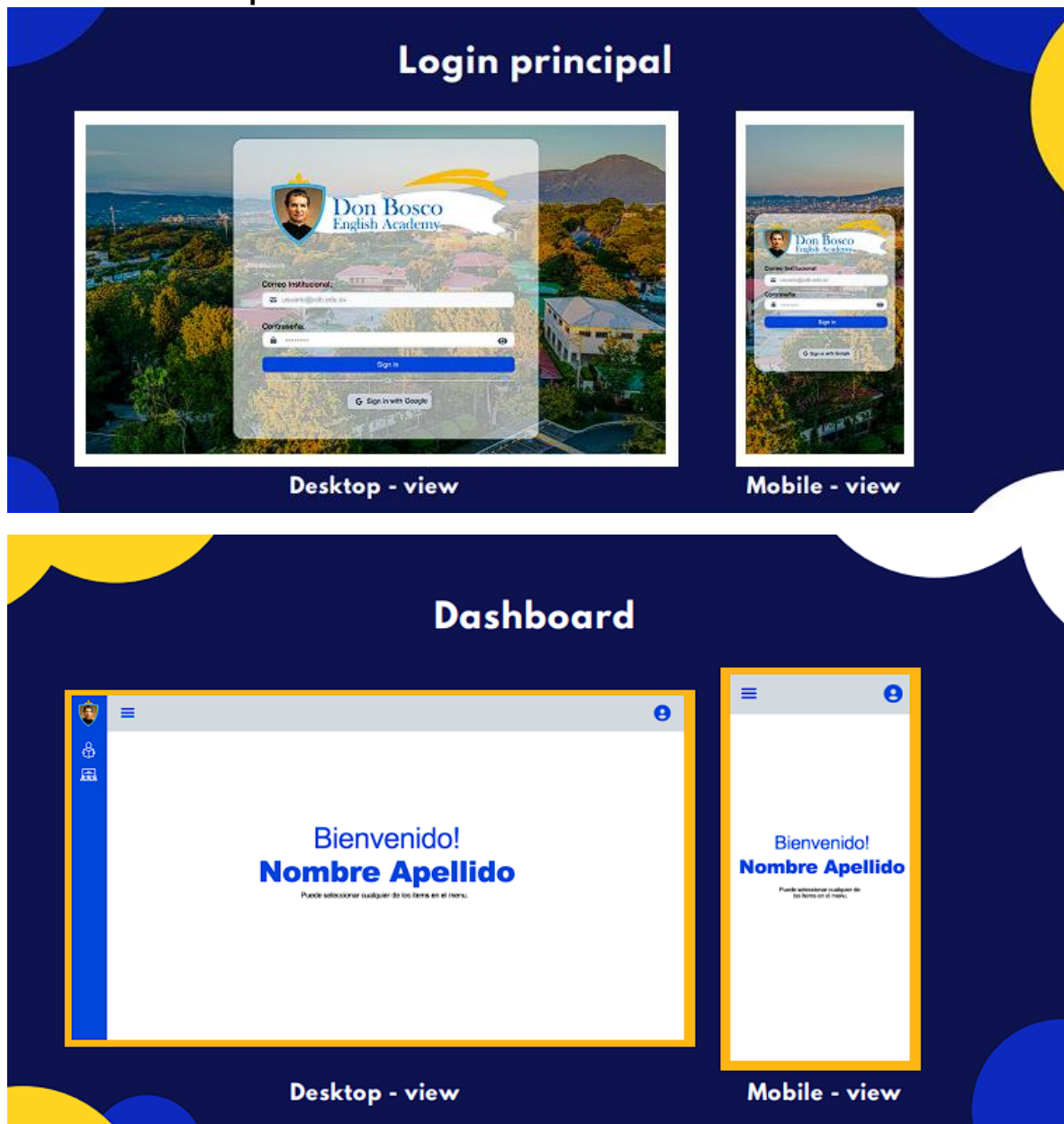
<b>validarCodigo</b>	Codigo	Función que devuelve un valor verdadero si el código ya existe y falso si el código que recibe no existe, esto con el objetivo de validar que el código sea único, en dado caso el resultado es verdadero la función "generarCodigo" se vuelve a ejecutar.
<b>incrementarCadena</b>	codigoModulo	Se utiliza para genera el código del nuevo modulo a crear, ya que estos van de manera ascendente, por lo que debe de seguir la secuencia.

#### 1.4. Validaciones

En el apartado de validaciones se realizan con JQuery al momento de mandar a llamar las funciones de agregado de cualquier información, se obtiene la información introducida dentro del formulario, para así ser procesada y validada, verificando que la información no esté vacía en dado caso sea obligatoria y por medio de expresiones regulares se verifica que el formato de información sea el correcto según las necesidades de cada dato.

## 2. Diseño de la interfaz de usuario

### 2.1. Diseño de pantallas



## Listado de Estudiantes

Nombres	Apellidos	Correo encargado	N° FAC	Fecha FAC
Yahr Stewart	Sibrian Arista	yahrstewart16@gmail.com	FAC12345	XXXX/XXXX
Julian Alejandro	Mendez Arvelo	alejandrx23.am20@gmail.com	FBC24681	XX/XX/XXXX
Kennel Adonay	Vargas Hernandez	kennayred@gmail.com	ABC54321	XXXX/XXXX
Pedro Roberto	Alexardo Mendez	pediWen@gmail.com	ZEP82465	XXXX/XXXX
Paula Xiomara	Cabrera Ortiz	xcvCabreraOrg@gmail.com	POV12345	XXXX/XXXX

Showing 1 to 5 of 100 Entries

Desktop - view

encargado	N° FAC	Fecha FAC		
yd@gmail.com	FAC12345	XXXX/XXXX	✓	✗
jd@gmail.com	FBC24681	XXXX/XXXX	✓	✗
kd@gmail.com	ABC54321	XXXX/XXXX	✓	✗
md@gmail.com	ZEP82465	XXXX/XXXX	✓	✗
od@gmail.com	POV12345	XXXX/XXXX	✓	✗

Showing 1 to 5 of 100 Entries

Mobile - view

## Agregar Estudiante

**Nuevo Estudiantes**

Nombres: \*

Apellidos: \*

Edad: \*  Nivel: \*

Encargado: \*  Parentesco: \*

Modulo: \*  N° de Factura: \*

Correo: \*

Desktop - view



## Editar Estudiante

**Editar Estudiantes**

Nombres: \*  
Yahir Stewart

Apellidos: \*  
Sibrian Arriola

Edad: \*  
15

Nivel: \*  
Juvenil

Modulo: \*  
Modulo 10

N° de Factura: \*  
FAC12345

Encargado: \*  
Flor Arriola

Parentesco: \*  
Madre

Correo: \*  
yahirstewart19@gmail.com

Desktop - view

## Modulos por Nivel

**Modulos por Nivel**

Infantil Juvenil

Codigo	Nombre	Cantidad Alumnos	N° de Secciones	
MF1	Modulo 1	30	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF2	Modulo 2	22	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF3	Modulo 3	25	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF4	Modulo 4	20	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF5	Modulo 5	30	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>

Showing 1 to 5 of 27 Entries

Desktop - view

**Modulos por Nivel**

Infantil Juvenil

Codigo	Nombre	Cantidad Alumnos	N° de Secciones	
MF1	Modulo 1	30	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF2	Modulo 2	22	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF3	Modulo 3	25	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF4	Modulo 4	20	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
MF5	Modulo 5	30	2	<input type="radio"/> <input checked="" type="checkbox"/> <input type="checkbox"/>

Showing 1 to 5 of 27 Entries

Mobile - view

## Modificar Nivel

The desktop view shows a modal form titled 'Modificar Nivel: MIF1'. It contains three input fields: 'Nombre' (with placeholder 'Digitar el nuevo nombre'), 'Cantidad de Estudiantes' (with placeholder 'Digitar cantidad de estudiantes'), and 'N° de secciones' (with placeholder 'Digitar la cantidad de secciones del nivel'). Below the inputs are 'Modificar' and 'Cancel' buttons. The background shows a table of modules.

Codigo	Nombre	Cantidad de Estudiantes	N° de secciones
MF1	Modulo 1	20	2
MF2	Modulo 2	20	2
MF3	Modulo 3	20	2
MF4	Modulo 4	20	2
MF5	Modulo 5	30	2

Desktop - view

The mobile view shows the same modal form, adapted for a smaller screen. The input fields and buttons are scaled down, and the background table is partially visible.

Mobile - view

## Modificar Nivel - Exitoso

The desktop view shows a success message modal with a blue checkmark icon and the text 'El nivel ha sido modificado correctamente.' Below the message is an 'Okay' button. The background shows the same table of modules as in the previous view.

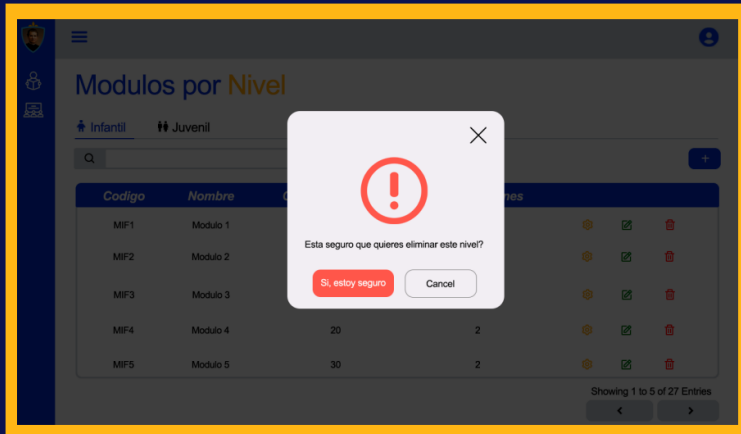
Codigo	Nombre	Cantidad de Estudiantes	N° de secciones
MF1	Modulo 1	20	2
MF2	Modulo 2	20	2
MF3	Modulo 3	20	2
MF4	Modulo 4	20	2
MF5	Modulo 5	30	2

Desktop - view

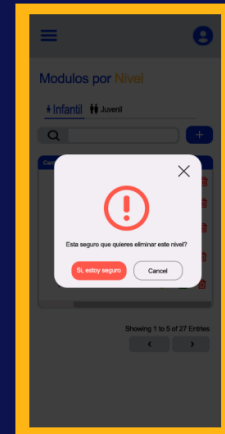
The mobile view shows the same success message modal, adapted for a smaller screen. The checkmark icon and text are scaled down, and the 'Okay' button is also scaled down.

Mobile - view

## Eliminar Nivel

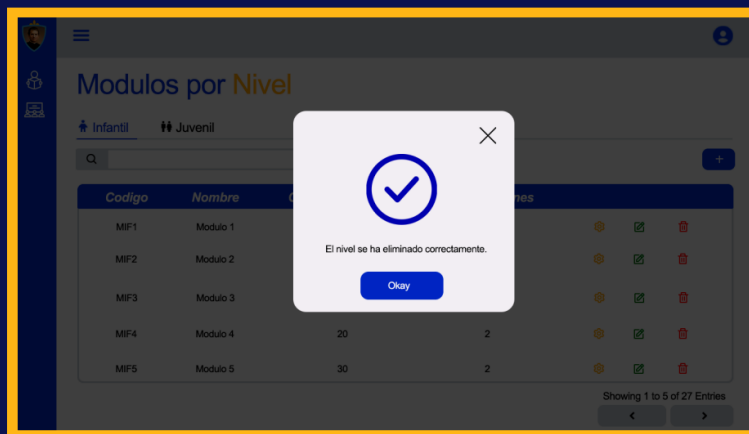


Desktop - view

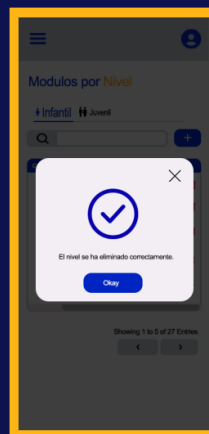


Mobile - view

## Eliminar Nivel - Exitosa



Desktop - view



Mobile - view

### 3. Elementos estáticos

Ingreso de Nuevo Estudiante: Este apartado se encuentra un formulario en el cual podemos introducir los datos de los nuevos estudiantes que entrarán en la academia, cuyos datos serán proporcionados por los padres del estudiante.

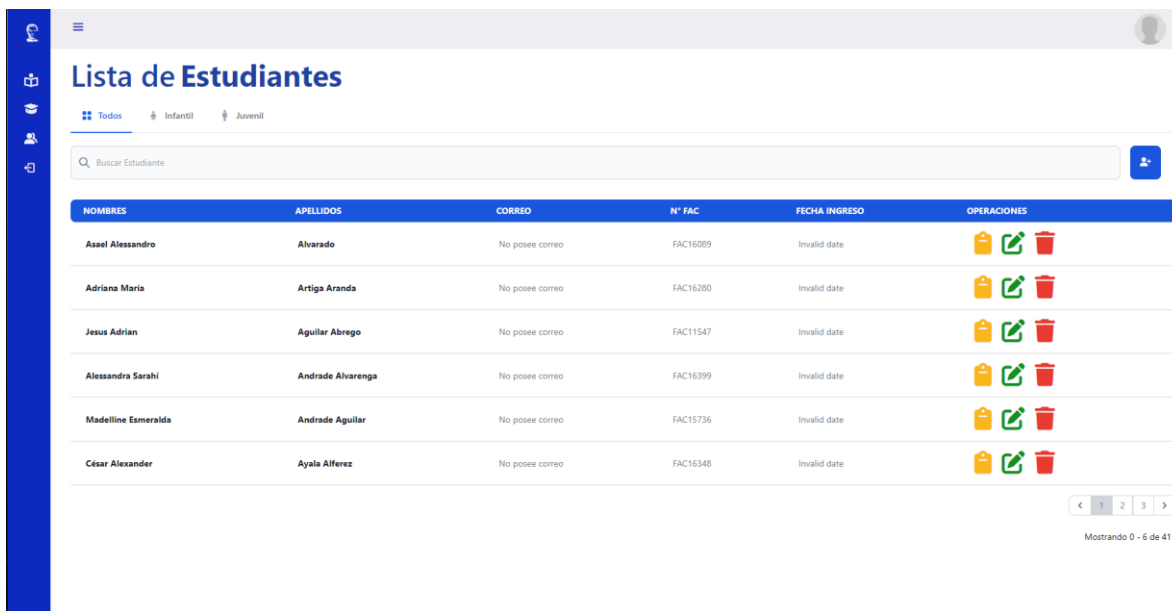
The screenshot shows a web application interface for adding a new student. On the left is a blue sidebar with icons for home, add, school, user, and a menu. The main header is light purple with a hamburger menu icon and a user profile icon. The title 'Nuevo Estudiante' is in blue. The form contains several fields and buttons:



















- Nombres:** A text input field with the placeholder 'Escribir nombres'.
- Apellidos:** A text input field with the placeholder 'Escribir apellidos'.
- Edad:** A text input field with the placeholder 'Escribir edad'.
- Programa:** A blue button labeled 'Seleccionar Programa'.
- Módulo:** A blue button labeled 'Seleccionar Módulo'.
- N° de Factura:** A text input field with the placeholder 'Escribir numero de la factura'.
- Encargado:** A text input field with the placeholder 'Escribir nombre del encargado'.
- Parentesco:** A blue button labeled 'Seleccionar Parentesco'.
- Correo:** A text input field with the placeholder 'Escribir correo'.
- Telefono del Encargado:** A text input field with the placeholder 'Escribir numero'.
- Sexo del Estudiante:** A blue button labeled 'Seleccionar sexo'.
- Estudiante CDB:** A blue button labeled 'Seleccionar respuesta'.

At the bottom left of the form area are two small square buttons: a red one with a white 'X' and a green one with a white checkmark.

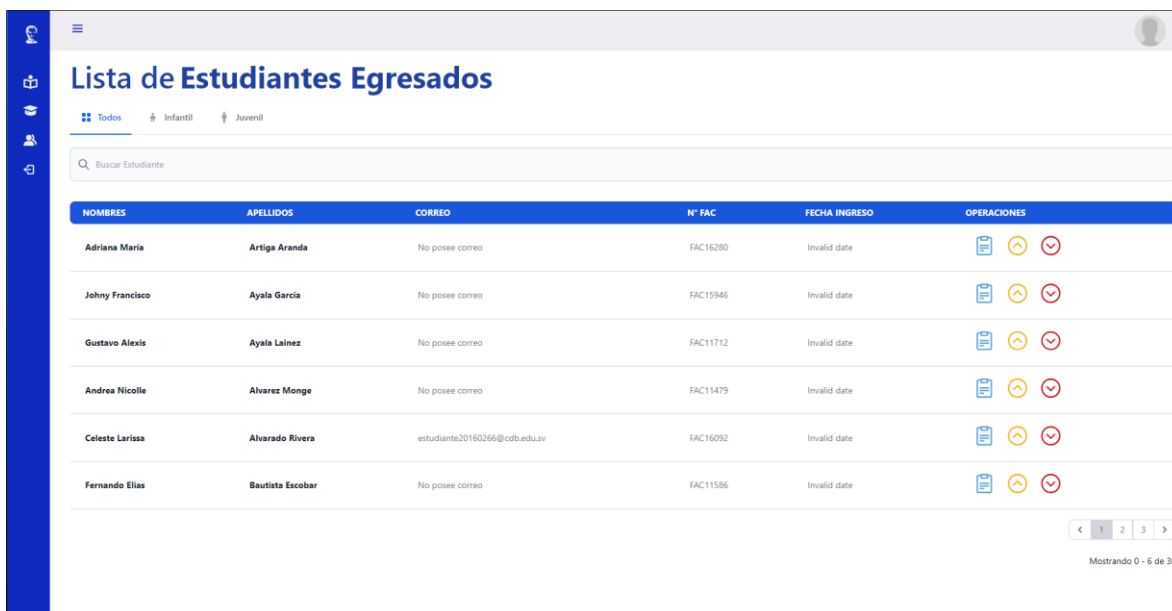
## 4. Elementos dinámicos



















Lista de Estudiantes: En el siguiente caso, podemos observar la presentación de los estudiantes en general inscritos dentro de la plataforma. Se compone de una tabla con los principales datos a conocer de los inscritos, cada fila con sus botones de acciones (ver información completa, modificar los datos y eliminar). A la vez incluye una tabulación para hacer de manera dinámica, ordenada y estética los listados.



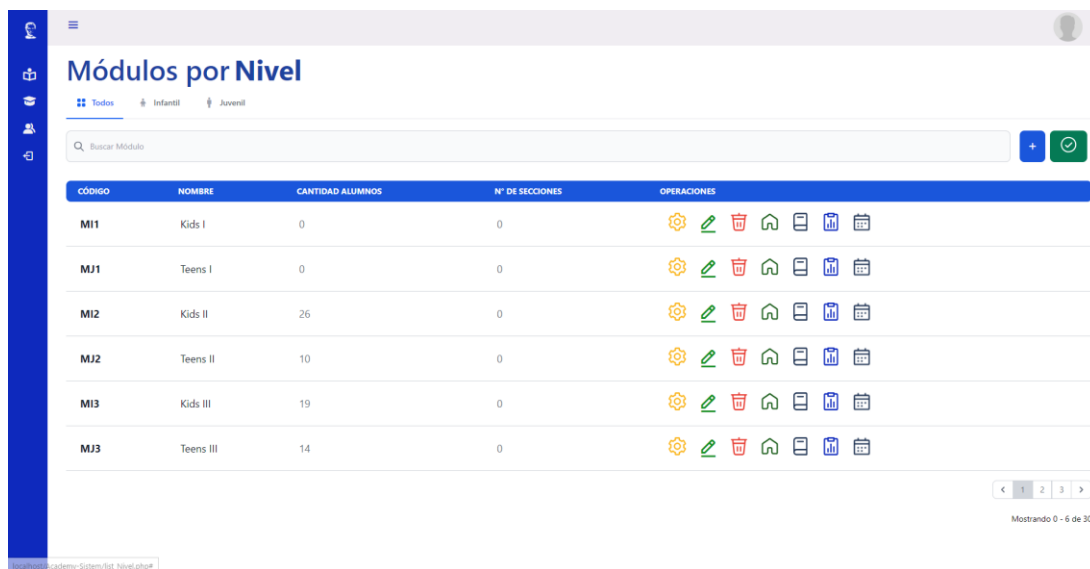
NOMBRES	APELLIDOS	CORREO	N° FAC	FECHA INGRESO	OPERACIONES
Asael Alessandro	Alvarado	No posee correo	FAC16089	Invalid date	  
Adriana Maria	Artiga Aranda	No posee correo	FAC16280	Invalid date	  
Jesus Adrian	Aguilar Alvrego	No posee correo	FAC11547	Invalid date	  
Alessandra Sarahi	Andrade Alvarenga	No posee correo	FAC16399	Invalid date	  
Madelline Esmeralda	Andrade Aguilar	No posee correo	FAC15736	Invalid date	  
César Alexander	Ayala Alferez	No posee correo	FAC16348	Invalid date	  

Lista de Estudiantes Egresados: De manera similar que con la presentación de los listados generales de los estudiantes, podemos observar la presentación de los estudiantes egresados. Con una composición similar, una tabla con los principales datos a conocer del egresado, cada fila con sus botones de acciones (ver información completa, modificar los datos y eliminar). A la vez incluye una tabulación para hacer de manera dinámica, ordenada y estética los listados.



NOMBRES	APELLIDOS	CORREO	N° FAC	FECHA INGRESO	OPERACIONES
Adriana Maria	Artiga Aranda	No posee correo	FAC16280	Invalid date	  
Johny Francisco	Ayala Garcia	No posee correo	FAC15946	Invalid date	  
Gustavo Alexis	Ayala Lainez	No posee correo	FAC11712	Invalid date	  
Andrea Nicolle	Alvarez Monge	No posee correo	FAC11479	Invalid date	  
Celeste Larissa	Alvarado Rivera	estudiante20160266@cib.edu.sv	FAC16092	Invalid date	  
Fernando Elias	Bautista Escobar	No posee correo	FAC11586	Invalid date	  

Módulos por Nivel: Se presentan los módulos que han sido registrados por la coordinación para la habilitación de los mismos, en cada uno se presentan los botones para interactuar con ellos.

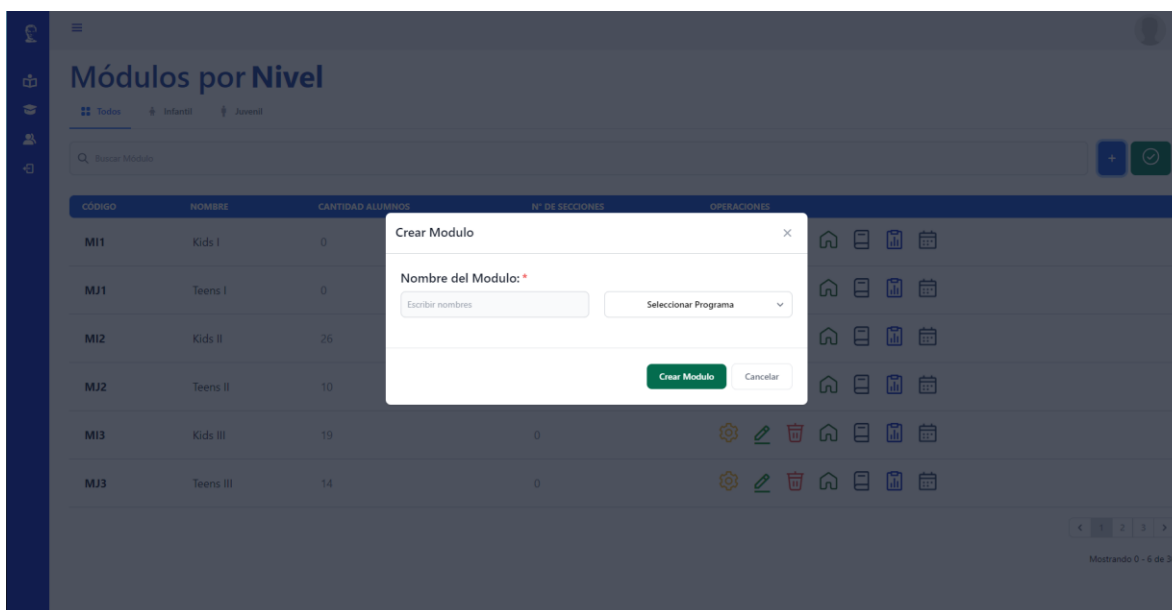


The screenshot shows a web application interface titled "Módulos por Nivel". It features a sidebar on the left with navigation icons. The main content area has a search bar and a table with the following data:

CÓDIGO	NOMBRE	CANTIDAD ALUMNOS	N° DE SECCIONES	OPERACIONES
MI1	Kids I	0	0	[Icons: Settings, Edit, Delete, Home, Document, Calendar]
MJ1	Teens I	0	0	[Icons: Settings, Edit, Delete, Home, Document, Calendar]
MI2	Kids II	26	0	[Icons: Settings, Edit, Delete, Home, Document, Calendar]
MJ2	Teens II	10	0	[Icons: Settings, Edit, Delete, Home, Document, Calendar]
MI3	Kids III	19	0	[Icons: Settings, Edit, Delete, Home, Document, Calendar]
MJ3	Teens III	14	0	[Icons: Settings, Edit, Delete, Home, Document, Calendar]

At the bottom right, there is a pagination control showing "Mostrando 0 - 6 de 30".

En este modal, podemos colocar el nombre que deseamos colocarle y el tipo de programa que seguirá el nuevo modulo para su atención.

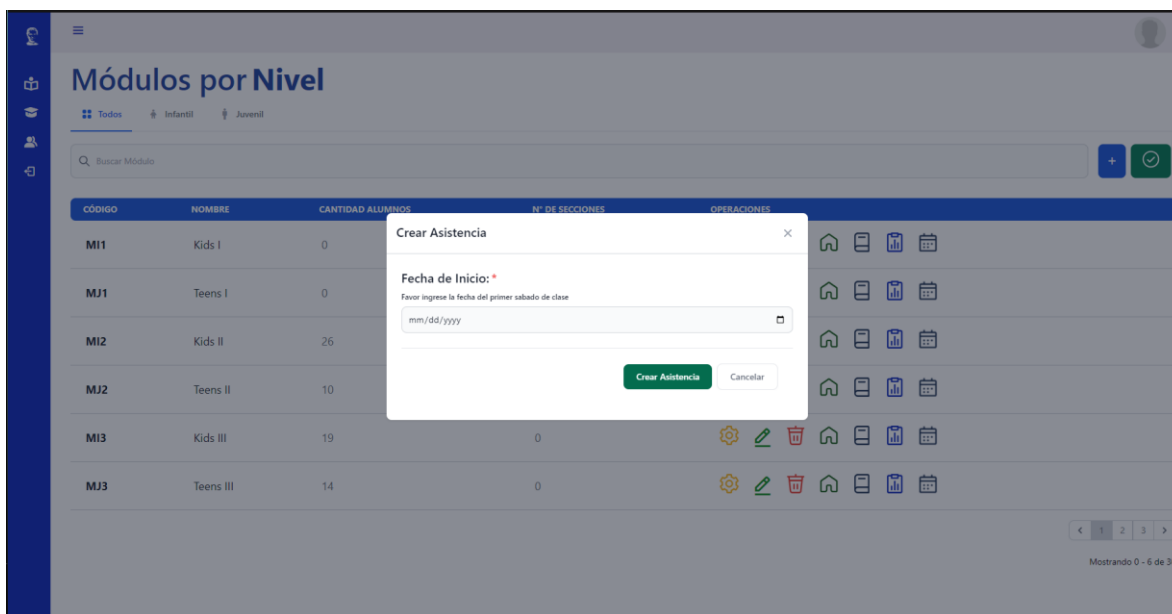


The screenshot shows the same "Módulos por Nivel" interface, but with a "Crear Modulo" modal open. The modal contains the following fields and buttons:

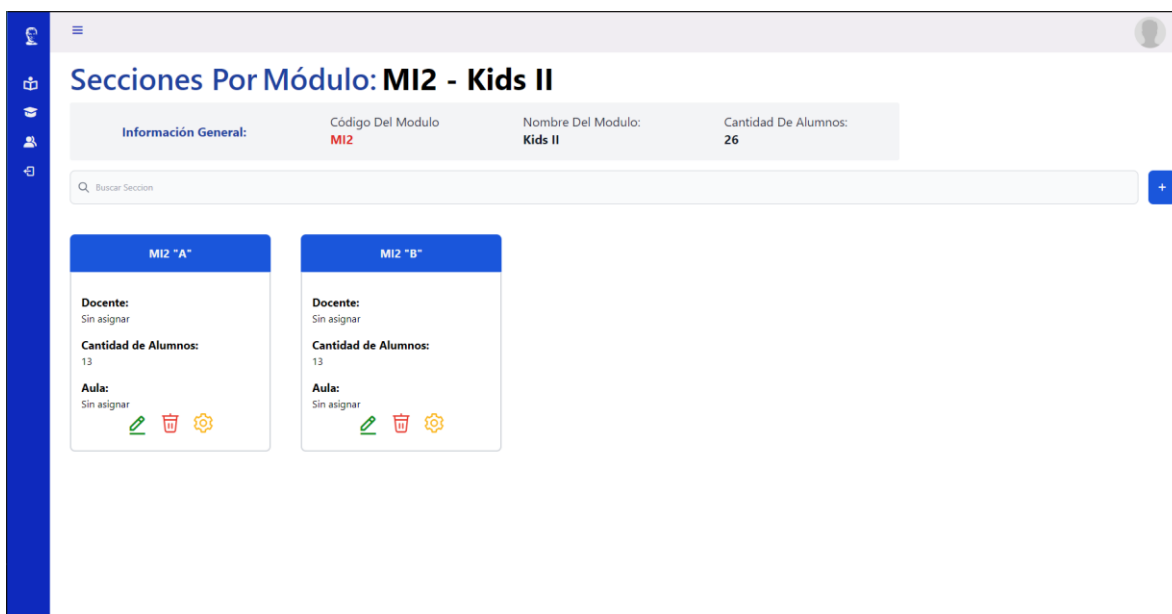
- Nombre del Modulo: \*
- Input field: Escribir nombres
- Dropdown menu: Seleccionar Programa
- Buttons: Crear Modulo, Cancelar

The background table and sidebar are dimmed.

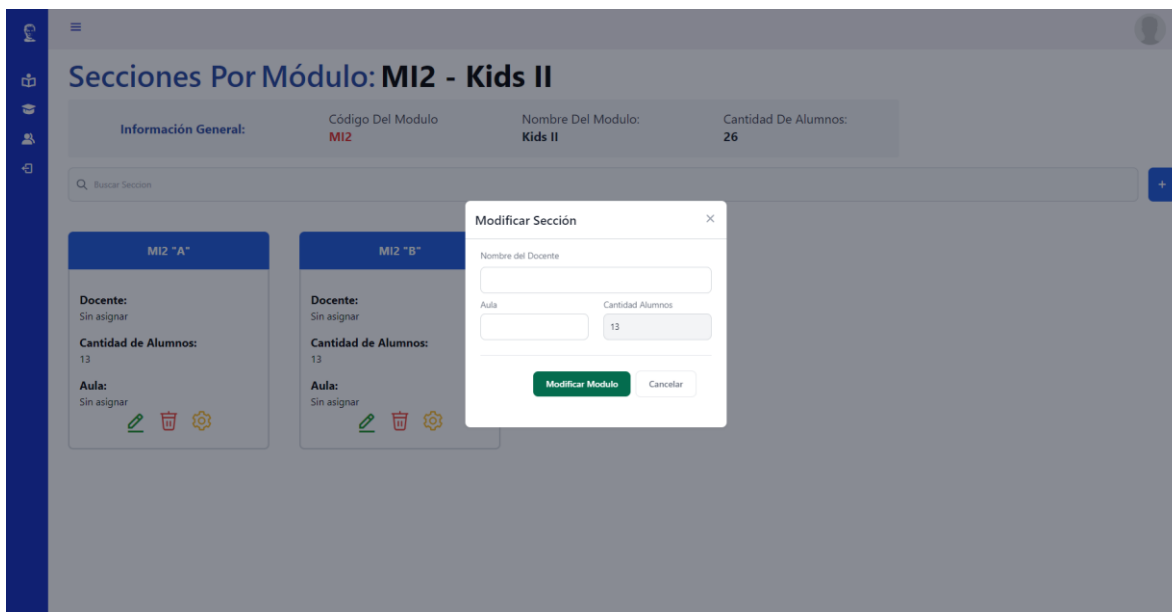
En el siguiente modal se selecciona la fecha a partir de cuando iniciará el módulo para así hacer de manera automática el cálculo de 8 sábados para crear los listados de asistencia.



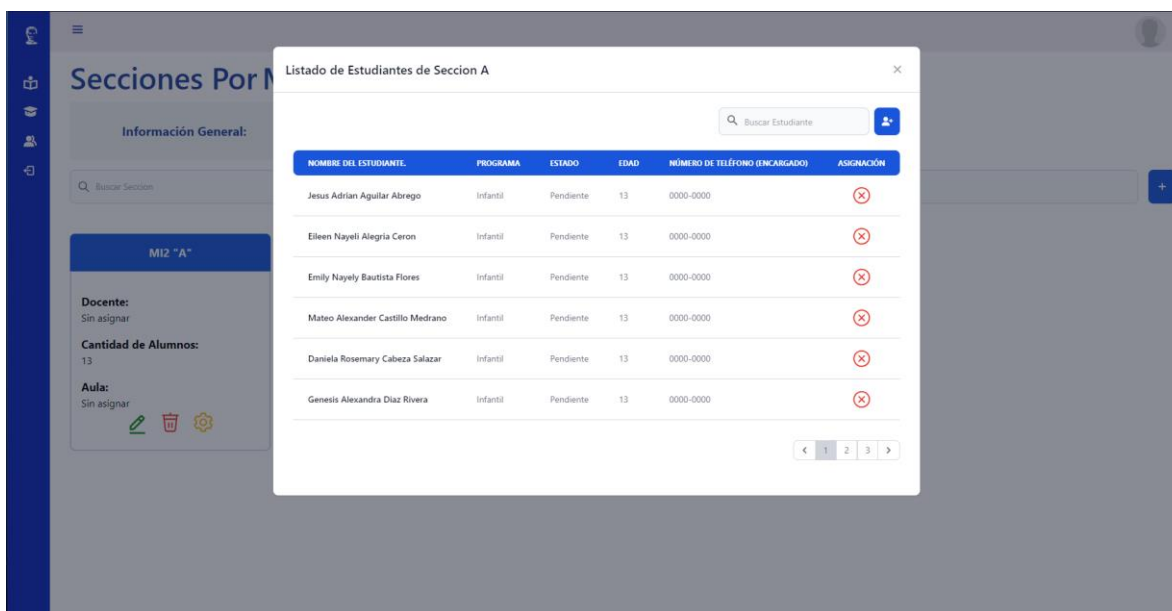
Selección por Módulos: Con esta vista, nosotros podremos observar las secciones en las que está formado el módulo. Cada sección cuenta con ventanas modales que se despliegan con los botones de acción.



Por ejemplo en este modal podemos modificar el nombre del docente a cargo y el aula en el que recibirán las clases los alumnos de dicha sección.



En el modal que se presenta, podemos ver los estudiantes de cada sección.





## 5. Temas y estilos

El estilo de la interfaz gráfica es bastante simple e intuitiva, ya que fue solicitado por hacerlo lo más entendible para el personal, se usa un tamaño de fuente bastante considerable para que no sea difícil de leer, además que los mensajes de confirmación o de alerta no se muestra con la función que proporciona por defecto los navegadores, sino que se crea un modal para poder mostrar la notificación para que sea más fácil de saber que es lo que sucedió

### Terminología de colores



## 6. Creación de las fuentes de datos y objetos

Para el estudiante posterior de haber recolectado toda la información y haber sido pasada al modelo se recibe en la siguiente parte del código

```
if ($action == "agregar") {
    //CAPTURA DE DATOS DEL AJAX
    $nombres = ($_POST['nombres']) ? $_POST['nombres'] : "";
    $apellidos = ($_POST['apellidos']) ? $_POST['apellidos'] : "";
    $edad = ($_POST['edad']) ? $_POST['edad'] : "";
    $factura = ($_POST['factura']) ? $_POST['factura'] : "";
    $encargado = ($_POST['encargado']) ? $_POST['encargado'] : "";
    $programa = ($_POST['programa']) ? $_POST['programa'] : "";
    $parentesco = ($_POST['parentesco']) ? $_POST['parentesco'] : "";
    $modulo = ($_POST['modulo']) ? $_POST['modulo'] : "";
    $correo = ($_POST['correo']) ? $_POST['correo'] : "";
    $telefono = ($_POST['telefono']) ? $_POST['telefono'] : "";
    $sexo = ($_POST['sexo']) ? $_POST['sexo'] : "";
    $alumnocdb = ($_POST['alumnocdb']) ? $_POST['alumnocdb'] : "";
    $codigocdb = ($_POST['codigocdb']) ? $_POST['codigocdb'] : "";
    date_default_timezone_set("America/El_Salvador");
    $fecha = date("Y-m-d");
    $estado = "";

    if($factura == "" || $factura == null){
        $estado = "Pendiente";
    } else {
        $estado = "Activo";
    }

    include_once("../recursos/consultas.php");
    do {
        $codigo = generarCodigo($apellidos);
        $verifycode = validarCodigo($codigo);
    } while ($verifycode == true);

    if($factura != "" || $factura != null){
        $fechafac = date("Y-m-d");
    }else{
        $fechafac = "0000-00-00";
    }

    //VARIABLES DE CONEXION
    $modelo = new Conexion;
    $db = $modelo->get_conexion();
    //echo ("<script>alert('Estas en el archivo de php, el nombre es:+'.$nombres.'')</script>");

    $stmt = $db->prepare("CALL add_estudiante (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
    $stmt->bindParam(1, $nombres, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $stmt->bindParam(2, $apellidos, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $stmt->bindParam(3, $edad, PDO::PARAM_INT | PDO::PARAM_INPUT_OUTPUT, 4000);
    $stmt->bindParam(4, $programa, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $stmt->bindParam(5, $factura, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $stmt->bindParam(6, $encargado, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
```

En esta parte se separan los datos a mandar al procedimiento almacenado, además de generar el código del estudiante, para poner identificarlo dentro del sistema, su estado de acuerdo si hay o no número de factura y de igual manera la fecha si hay o no hay numero de factura.

## Actualizar información de sección

```
$aula = ($_POST['aula']) ? $_POST['aula'] : "";
$docente = ($_POST['docente']) ? $_POST['docente'] : "";
$modulo = ($_POST['modulo']) ? $_POST['modulo'] : "";
$seccion = ($_POST['seccion']) ? $_POST['seccion'] : "";

$db = new Conexion();
$modelo = $db->get_conexion();

if($modelo){
    $update = $modelo->prepare("CALL update_seccion(?,?,?,?)");
    $update->bindParam(1,$aula, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $update->bindParam(2,$docente, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $update->bindParam(3,$modulo, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $update->bindParam(4,$seccion, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);

    $update->execute();

    $count = $update->rowCount();

    if($count>0){
        $json['status'] = true;
        $json['msg']="Actualización exitosa";
    } else {
        $json['status']=false;
        $json['msg'] = "No se pudo actualizar";
    }
} else {
    $json['status']=false;
    $json['msg']="No hay conexion";
}

echo json_encode($json);
}
```

En este apartado se recogen los datos que se necesitan para poder agregar la información a la sección, estos datos son previamente validados por parte del JQuery.

## Agregar modulo

```
if($action == "agregar"){
    $nombres = ($_POST['nombre'])?$_POST['nombre']:"";
    $programa = ($_POST['programa'])?$_POST['programa']:"";
    $letras="";

    include_once('../recursos/consultas.php');

    if($programa == "Infantil"){
        $lista = getCodigos($programa);
        $letras = "MI";
    } else if($programa == "Juvenil"){
        $lista = getCodigos($programa);
        $letras = "MJ";
    }

    if(empty($lista)){
        $num = 1;
    } else {
        $num = extraerLast(array_pop($lista));
    }

    $codigo = $letras . $num;

    $db = new Conexion();
    $modelo = $db->get_conexion();

    $stmt = $modelo->prepare("CALL add_modulo (?, ?, ?)");
    $stmt->bindParam(1,$nombres, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $stmt->bindParam(2,$codigo, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);
    $stmt->bindParam(3,$programa, PDO::PARAM_STR | PDO::PARAM_INPUT_OUTPUT, 4000);

    $stmt->execute();
    $cont = $stmt->rowCount();

    if($cont>0){
        $json['status']=true;
        $json['msg']="Se ha agregado correctamente ".$codigo;
    } else {
        $json['status']=true;
        $json['msg']="Ha ocurrido un error";
    }
    echo json_encode($json);
}
```

Apartado donde se recolecta la información del modulo a crear, además que se crea el siguiente código del módulo a partir del programa, y se avanza uno a partir de los módulos anteriormente creados.

## 7. Programación de algunas clases, procedimientos funciones y validaciones

### 7.1.Procedimientos

A manera de ejemplo tenemos el procedimiento almacenado de “mostrar” hay un pequeño menú que de acuerdo con el texto que nosotros mandemos hará selección de SELECT que se necesita para ese preciso momento, así solo necesitando la llamada al procedimiento almacenado y la variable especificando que datos deseamos mostrar.

The screenshot shows the configuration for a stored procedure named 'mostrar'. It has one input parameter 'op' of type VARCHAR with a length of 30. The procedure body is as follows:

```
1 IF op = 'estudiante' THEN
2   SELECT * FROM estudiante;
3 ELSEIF op = 'modulo' THEN
4   SELECT * FROM modulo ORDER BY CAST(SUBSTRING(id_modulo, 3) AS UNSIGNED);
5   ELSEIF op = 'listaM' THEN
6     SELECT id_modulo FROM modulo ORDER BY CAST(SUBSTRING(id_modulo, 3) AS UNSIGNED);
7   ELSEIF op = 'infantil' THEN
8     SELECT * FROM estudiante WHERE programa = 'Infantil';
9   ELSEIF op = 'juvenil' THEN
10    SELECT * FROM estudiante WHERE programa = 'Juvenil';
11  ELSEIF op = 'MInfantil' THEN
12    SELECT * FROM modulo WHERE programa = 'Infantil';
13  ELSEIF op = 'MJuvenil' THEN
14    SELECT * FROM modulo WHERE programa = 'Juvenil';
15  ELSEIF op = 'egresados' THEN
16    SELECT * FROM estudiante WHERE estado = 'Egresado';
```

Como otro ejemplo tenemos el procedimiento almacenado de “deletes”, donde se almacenan todas las funciones para poder borrar cualquier elemento que se haya contemplado, se necesitan enviar dos parámetros el primero donde especificamos que deseamos borrar y el segundo parámetro sería el ID del elemento a borrar.

The screenshot shows the configuration for a stored procedure named 'deletes'. It has two input parameters: 'op' of type VARCHAR with a length of 20, and 'id' of type VARCHAR with a length of 10. The procedure body is as follows:

```
1 IF op = 'estudiante' THEN
2   DELETE FROM estudiante WHERE id_estudiante = id;
3 ELSEIF op = 'modulo' THEN
4   DELETE FROM modulo WHERE id_modulo = id;
5 END IF
```

De igual manera, el procedimiento "show\_individual" según la opción que nosotros le pasemos como parámetros se ejecutará una consulta en específico para ver la información de un solo estudiantes o los estudiantes que pertenecen a cierto modulo en específico.

PROCEDURE ▾

	Direction	Name	Type	Length/Values	Options	
⬆	IN ▾	<input type="text" value="op"/>	VARCHAR ▾	<input type="text" value="30"/>	Charset ▾	⊖ Drop
⬆	IN ▾	<input type="text" value="id"/>	VARCHAR ▾	<input type="text" value="20"/>	Charset ▾	⊖ Drop

Add parameter

```
1 IF op = 'estudiante' THEN
2   SELECT * FROM estudiante WHERE id_estudiante = id;
3 ELSEIF op = 'modulo' THEN
4   SELECT * FROM modulo WHERE id_modulo = id;
5 ELSEIF op = 'porModulo' THEN
6   SELECT * FROM estudiante WHERE modulo = id;
7   END IF
```

## 7.2. Funciones

## Función agregarEstudiante

[illegible]

Luego de haber recuperado y validado toda la información se realiza la función Ajax donde se agrega el estudiante, mandando la acción al modelo junto a la información recuperada, ejecutando el procedimiento almacenado para guardar los datos.

## Función mostrarEstudiante

```
if ($("#alumnos").length) {  
  mostrarEstudiante(1);  
}  
  
function mostrarEstudiante(pagina) {  
  let action = "mostrar";  
  
  // Cantidad de elementos a mostrar por página  
  var elementosPorPagina = 6;  
  
  $.ajax({  
    type: "POST",  
    url: "modelo/class.estudiante.php",  
    data: {  
      action: action,  
    },  
    success: function (response) {  
      var estudiantes = JSON.parse(response);  
      var resultado = "";  
      var factura;  
      var correo;  
  
      var mensaje = estudiantes.status;  
      if (estudiantes.status == false) {  
        resultado = "<td colspan='6' style='text-align: center; font-size:15px;'><b><h1 class='font-sans py-4'>" + estudiantes.msg + "</h1></b></td>";  
      } else {  
        // Cálculo de los índices de inicio y fin para la página actual  
        var inicio = (pagina - 1) * elementosPorPagina;  
        var fin = inicio + elementosPorPagina;  
        estudiantes.data.slice(inicio, fin).forEach((estudiante) => {  
          if (estudiante.num_factura == "") {  
            factura = "No posee factura";  
          } else {  
            factura = estudiante.num_factura;  
          }  
  
          if (estudiante.correo == "") {  
            correo = "No posee correo";  
          } else {  
            correo = estudiante.correo;  
          }  
  
          var fecha;  
          fecha = moment(estudiante.fecha_ingreso).format("DD/MM/YYYY");  
          resultado += `  
            <tr class="bg-white border-b dark:bg-gray-300 dark:border-gray-300 overscroll-x-none">  
              <th class="font-sans px-6 py-4 text-sm text-gray-900 whitespace-nowrap dark:text-white">  
                ${estudiante.nombres}  
              </th>  
              <th class="font-sans px-6 py-4 text-sm text-gray-900 whitespace-nowrap dark:text-white">  
                ${estudiante.apellidos}  
              </th>  
              <td class="px-6 py-4 font-sans text-sm">  
                ${correo}  
              </td>  
              <td class="px-6 py-4 font-sans text-sm">  
                ${factura}  
              </td>  
            `;  
        });  
      }  
    }  
  });  
}
```

Función con una petición Ajax, donde se recuperan toda la información de los estudiantes para poder presentarla dentro del aplicativo web, de una manera similar o igual trabajan las demás funciones de mostrar dentro del aplicativo.



## Función info

```
$(document).on("click", "#info", function () {
    var estu = $(this).attr("data_id");
    var action = "individual";
    var resultado = "";
    var factura = "";
    $.ajax({
        type: "POST",
        url: "modelo/class.estudiante.php",
        data: {
            action: action,
            id: estu,
        },
        success: function (response) {
            var alumno = JSON.parse(response);
            var modulo = "";
            var correo = "";
            var factura = "";
            var codigocdb = "";
            if (alumno.status == false) {
                resultado = alumno.msg;
            } else {
                alumno.data.forEach((alum) => {
                    if (alum.modulo == "") {
                        modulo = "Sin asignar";
                    } else {
                        modulo = alum.modulo;
                    }
                });
                if (alum.correo == "") {
                    correo = "No posee";
                } else {
                    correo = alum.correo;
                }
                if (alum.num_factura == "") {
                    factura = "No posee";
                } else {
                    factura = alum.num_factura;
                }
                if (alum.codigo_cdb == "") {
                    codigocdb = "No posee";
                } else {
                    codigocdb = alum.codigo_cdb;
                }
                estado = alum.estado;
                resultado += `
```

Función para mostrar la información de un estudiante en específico, donde se hace la solicitud Ajax con su acción para imprimir un modal con la información de este estudiante.

## Función generarCodigo

```
1 reference
function generarCodigo($ape){
    $primeraletra = substr($ape,0,1);
    $segundalettra = strpos($ape, ' ');
    $second = substr($ape, $segundalettra + 1,1);
    $num = sprintf("%03d", rand(0,999));
    $codigo = $primeraletra . $second . $num;

    return $codigo;
}
```

Función de php para generar el código de manera automática, a partir de las primeras letras de su apellido o apellidos, agregando tres números aleatorios.

## Función validarCodigo

```
1 reference
function validarCodigo($code){
    $db = new Conexion();
    $dbh = $db->get_conexion();
    $sql = "SELECT * FROM estudiante WHERE Id_estudiante = :code";
    $stmt = $dbh->prepare($sql);
    $stmt ->bindParam(':code', $code);
    $stmt->execute();

    if($stmt->rowCount()){
        return true;
    } else {
        return false;
    }
}
```

Función para validar la existencia del código de un estudiante con anterioridad, esto para validar que la creación del código del estudiante sea única dentro de la base de datos.

## Función incrementarCadena

```
//PARA INCREMENTAR EL MODULO AL QUE VA A IR
1 reference
function incrementarCadena($cadena) {
    // Extraer el prefijo y el número de la cadena
    $prefijo = substr($cadena, 0, 2);
    $numero = (int) substr($cadena, 2);

    // Incrementar el número en 1
    $numero++;

    // Combinar el prefijo con el nuevo número
    $nuevaCadena = $prefijo . $numero;

    return $nuevaCadena;
}
```

Función para incrementar por 1 el siguiente modulo que se va a crear.

### 7.3. Validaciones

En el tema de validaciones se realiza antes de agregar el estudiante por medio de JQuery, donde se obtienen los datos de manera directa del formulario y por medio de expresiones regulares se validan ciertos formatos que se necesitan, por ejemplo: el nombre que tenga solo letra, que la edad solo sean números, que el correo tenga un formato en específico y la factura de igual manera, en dado caso no cumple lo solicitado, el input se pondrá de color rojo indicando donde está el error y a su vez muestra una notificación que brinda el navegador, brindando información extra sobre el error y lo que se debe hacer para solucionarlo, de manera similar se realiza para toda entrada y guardado de datos que se tiene dentro del aplicativo.

```
//EXPRESIONES REGULARES
var texto = /^[A-ZÁÉÍÓÚa-zñáéíóúÁÉÍÓÚ\.[\s]+$/;
var nums = /^[1-9][0-9]?$/^100$/;
var regxcorreo = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
var regxtel= /[0-9]{4}\-[0-9]{4}/
var rexfac = /^FAC\d{4,6}$/;
//Variable de validacion para que todos los datos esten correctos
var val = true;
var correo = "";

//validacion del nombre
if($('#nombres').val()==" " || !texto.test($('#nombres').val())) {
    val = false;
    $('#nombres').css('border-color','red');
    alert("Favor revise el campo de nombre");
} else {
    $('#nombres').css('border-color','#D1D5DB');
}

//validacion de apellidos
if($('#apellidos').val()==" " || !texto.test($('#apellidos').val())){
    val = false;
    $('#apellidos').css('border-color','red');
    alert("Favor revise el campo de apellidos");
} else {
    $('#apellidos').css('border-color','#D1D5DB');
}

if($('#factura').val()!=" "){
    if(!refac.test($('#factura').val())){
        val = false;
        $('#factura').css('border-color','red');
        alert("Favor revise el campo de factura");
    }
} else {
    $('#factura').css('border-color','#D1D5DB');
}

if($('#edad').val()==" " || !nums.test($('#edad').val())){
    val = false;
    $('#edad').css('border-color','red');
    alert("Favor revise el campo de edad");
} else if($('#edad').val() < 6){
    val = false;
    $('#edad').css('border-color','red');
    alert("la edad minima es de 6 años");
}
```

Otro ejemplo de validaciones dentro del aplicativo es:

Para agregar módulos:

```
$("#frmAddModulo").submit(function (e) {  
    let nombre = $('#nombre').val();  
    let programa = $('#programa').val();  
    var val = true;  
  
    if(nombre == "" || nombre == null){  
        $('#nombre').css("border-color", 'red');  
        alert("Revise el campo del nombre");  
        val = false;  
    }  
  
    if(programa == "" || programa == null){  
        $('#programa').css('border-color', 'red');  
        alert("Seleccione una opcion");  
        val = false;  
    }  
}
```

Para actualizar la información de una sección

```
//ACCION PARA ACTUALIZAR LA SECCION  
$(document).on("click", "#UpdateSeccion", function (e) {  
    e.preventDefault();  
    var docente = $("#docente").val();  
    var aula = $("#salon").val();  
    var action = "updateSeccion";  
    var modulo = $("#codigoM").text();  
    var seccion = $("#UpdateSeccion").attr("data_id");  
    var texto = /^[A-ZÁÉÍÓÚa-zñáéíóúÁÉÍÓÚ\.[\s]*+$/;  
    var validar = true;  
  
    if (docente == "" || !texto.test(docente)) {  
        alert("Favor revise el nombre del docente");  
        $("#docente").css("border-color", "red");  
        validar = false;  
    } else {  
        $("#docente").css("border-color", "#CBD5E0");  
    }  
  
    if (aula == "") {  
        alert("Favor revise el aula");  
        $("#salon").css("border-color", "red");  
        validar = false;  
    } else {  
        $("#salon").css("border-color", "#CBD5E0");  
    }  
}
```

## 8. Creación de algunos formularios web

### Formulario para nuevo estudiante

**Nuevo Estudiante**

Nombres: \*  
Escribir nombres

Apellidos: \*  
Escribir apellidos

Edad: \*  
Escribir edad

Programa: \*  
Seleccionar Programa

Módulo:  
Seleccionar Módulo

N° de Factura:  
Escribir numero de la factura

Encargado: \*  
Escribir nombre del encargado



Parentesco: \*  
Seleccionar Parentesco

Correo:  
Escribir correo

Telefono del Encargado: \*  
Escribir numero

Sexo del Estudiante: \*  
Seleccionar sexo

Estudiante CDB: \*  
Seleccionar respuesta



Formulario para agregar un nuevo estudiante dentro del sistema, en él se presenta y solicita toda la información que se les pide a los padres llenar acerca del estudiante, no toda la información dentro del formulario es obligatoria, ya que algunos estudiantes no tienen a disposición esa información en el momento.































### Formulario para agregar módulos

**Módulos por Nivel**

Tools | Infantil | Juvenil

Buscar Modulo

CODIGO	NOMBRE	CANTIDAD ALUMNOS	N° DE SECCIONES	OPERACIONES
MI1	Kids I	0		   
MJ1	Teens I	0		   
MI2	Kids II	26		   
MJ2	Teens II	10		   
MI3	Kids III	19	0	      
MJ3	Teens III	14	0	      

Mostrando 0 - 6 de 30

El formulario se presenta dentro de una ventana modal, donde se solicita el nombre que se le va a asignar al módulo, y a su vez se solicita el programa al que pertenece este módulo, para así diferenciarlo dentro del sistema y su código sea creado de acuerdo con el programa.

## Formulario para crear secciones

The screenshot displays a web interface titled "Secciones Por Módulo: MI2 - Kids II". At the top, there is a summary bar with the following information: "Información General:", "Código Del Módulo: MI2", "Nombre Del Módulo: Kids II", and "Cantidad De Alumnos: 26". Below this is a search bar labeled "Buscar Sección" and a blue "+" button. The main content area shows two section cards, "MI2 'A'" and "MI2 'B'", each with fields for "Docente: Sin asignar", "Cantidad de Alumnos: 13", and "Aula: Sin asignar". A modal window titled "Cantidad de Secciones" is open in the center, containing a text input field labeled "Digite la cantidad de secciones:" and two buttons: "Crear Sección" and "Cancelar".

Formulario donde se solicita la cantidad de secciones que se desean crear, se validará que en cada sección deba haber 10 estudiantes como mínimo para estas puedan ser creada.

## Formulario para modificar la información de la sección

This screenshot shows the same "Secciones Por Módulo: MI2 - Kids II" interface. The modal window is now titled "Modificar Sección". It contains three input fields: "Nombre del Docente", "Aula", and "Cantidad Alumnos" (which has the value "13" pre-filled). At the bottom of the modal are two buttons: "Modificar Módulo" and "Cancelar". The background section cards remain visible but are slightly dimmed.

En este formulario se modifica la información de la sección, agregando la información del docente que estará encargado de esa sección y el aula donde estará impartiendo las clases.

## Formulario para crear asistencia

The screenshot shows a web application interface with a table titled "Módulos por Nivel". The table has columns: CÓDIGO, NOMBRE, CANTIDAD ALUMNOS, N° DE SECCIONES, and OPERACIONES. A modal titled "Crear Asistencia" is open, prompting the user to enter the "Fecha de Inicio" (Start Date) in mm/dd/yyyy format. The modal includes "Crear Asistencia" and "Cancelar" buttons.

CÓDIGO	NOMBRE	CANTIDAD ALUMNOS	N° DE SECCIONES	OPERACIONES
MI1	Kids I	0		
MJ1	Teens I	0		
MI2	Kids II	26		
MJ2	Teens II	10		
MI3	Kids III	19	0	
MJ3	Teens III	14	0	

Formulario donde se solicita la fecha que inicia o inició el modulo que se está desarrollando, para así hacer una ficha con los 8 sábados que conforma el periodo de un modulo para ir marcando la respectiva asistencia de los estudiantes.

## 9. Actualización de cronogramas de actividades

