

**Sistemas de Planillas de Notas de los Estudiantes**

**Manual del Programador**

Versión: 0.1

Fecha: 24/08/2023

El software "Sistema de Planilla de Notas de los Estudiantes" y la documentación asociada incluida en este manual están protegidos por las leyes de derechos de autor [Digital Redesing] retiene todos los derechos de propiedad del software y la documentación

**HOJA DE CONTROL**

|  |  | | |
| --- | --- | --- | --- |
| **Nombre del Proyecto** | **Sistemas de Planillas de Notas de los Estudiantes** | | |
| **Entregable** | Manual de Programador | | |
| **Autor** | Dilan Samuel Gutierrez Munevar, Jose David Hernandez Garcia, Joahn Santiago Ramirez Bustos. | | |
| **Versión/Edición** |  | **Fecha Versión** | 08/11/2023 |
| **Aprobado por** |  | **Fecha Aprobación** | DD/MM/AAAA |
|  |  | **Nº Total de Páginas** |  |

**Índice**

[**Introducción 4**](#_heading=h.txtzyjbuu8y7)

[**Mapa del sistema 5**](#_heading=h.2et92p0)

[Modelo Lógico 5](#_heading=h.v1eyw4e8lk0u)

[**Descripción Del Sistema 6**](#_heading=h.1t3h5sf)

[Objetivo 6](#_heading=h.30j0zll)

[Funcionalidad 6](#_heading=h.3znysh7)

[Subsistema 6](#_heading=h.4d34og8)

[**Orden de Archivos y BD 8**](#_heading=h.hascfbpekckh)

[**Login(Explicación código): 10**](#_heading=h.qvvr9vgtt5qc)

[**Menú para diferentes usuarios 13**](#_heading=h.4w79gvqhub6q)

[**Variable de conexión, métodos y consulta utilizadas a lo largo del código: 15**](#_heading=h.l17t78bpgfxi)

[**Trigger 25**](#_heading=h.3aka0utiod20)

[**Glosario 30**](#_heading=h.35nkun2)

# Introducción

El presente documento describe los aspectos técnicos informáticos del sistema de

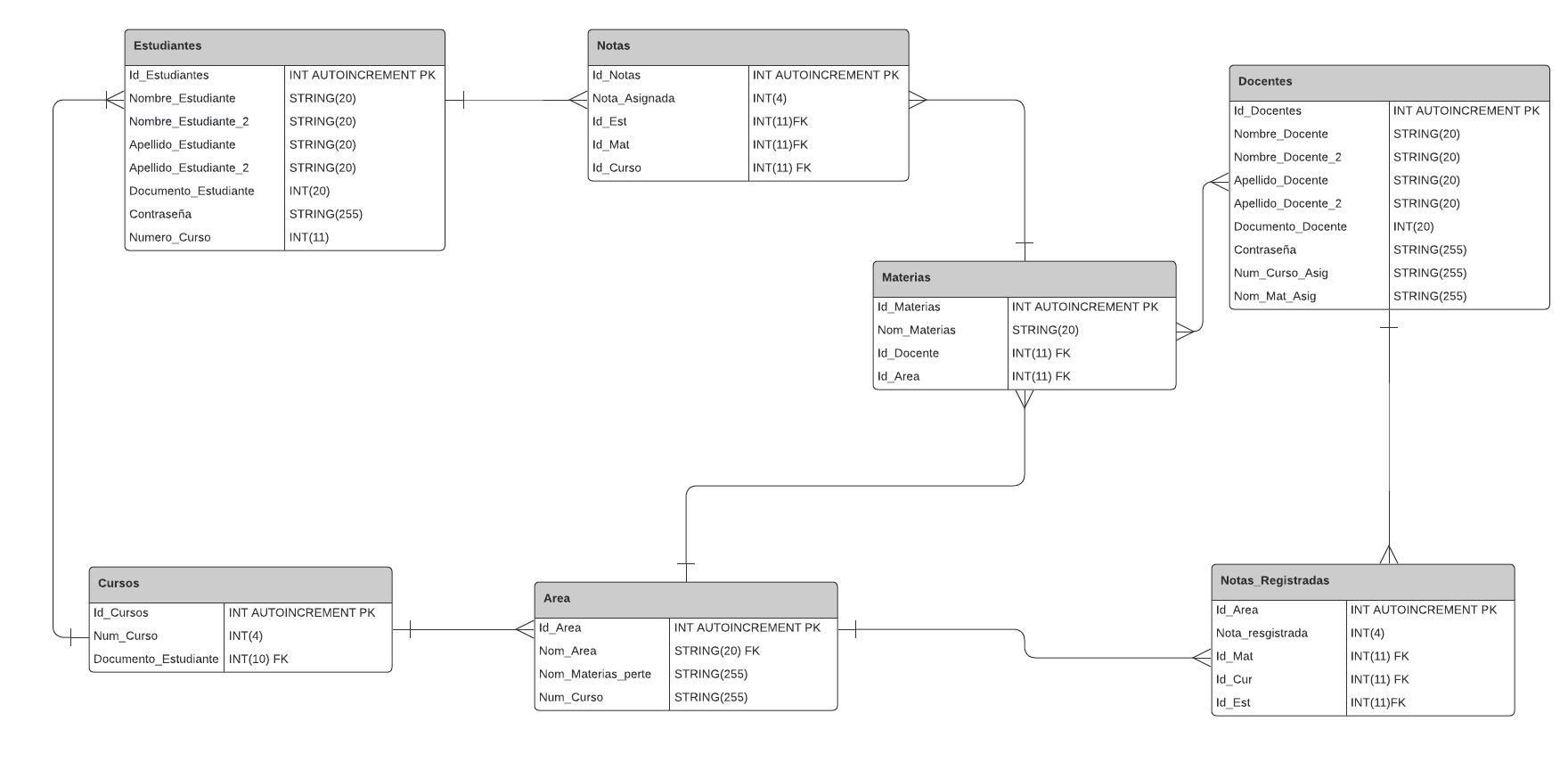
información. El documento familiariza al personal técnico especializado encargado de las

actividades de mantenimiento, revisión, solución de problemas, instalación y configuración

del sistema.

# Mapa del sistema

## Modelo Lógico



# Descripción Del Sistema

El Sistema de Planilla de Notas para Estudiantes es una plataforma integral diseñada para simplificar y automatizar el proceso de seguimiento y gestión de las calificaciones y el desempeño académico de los estudiantes de la institución. Este sistema altamente eficiente proporciona a educadores, administradores y estudiantes las herramientas para supervisar el rendimiento académico

## Objetivo

Crear un sistema de información para sistematizar los procesos de notas del Instituto Técnico Juan del Corral.

## Funcionalidad

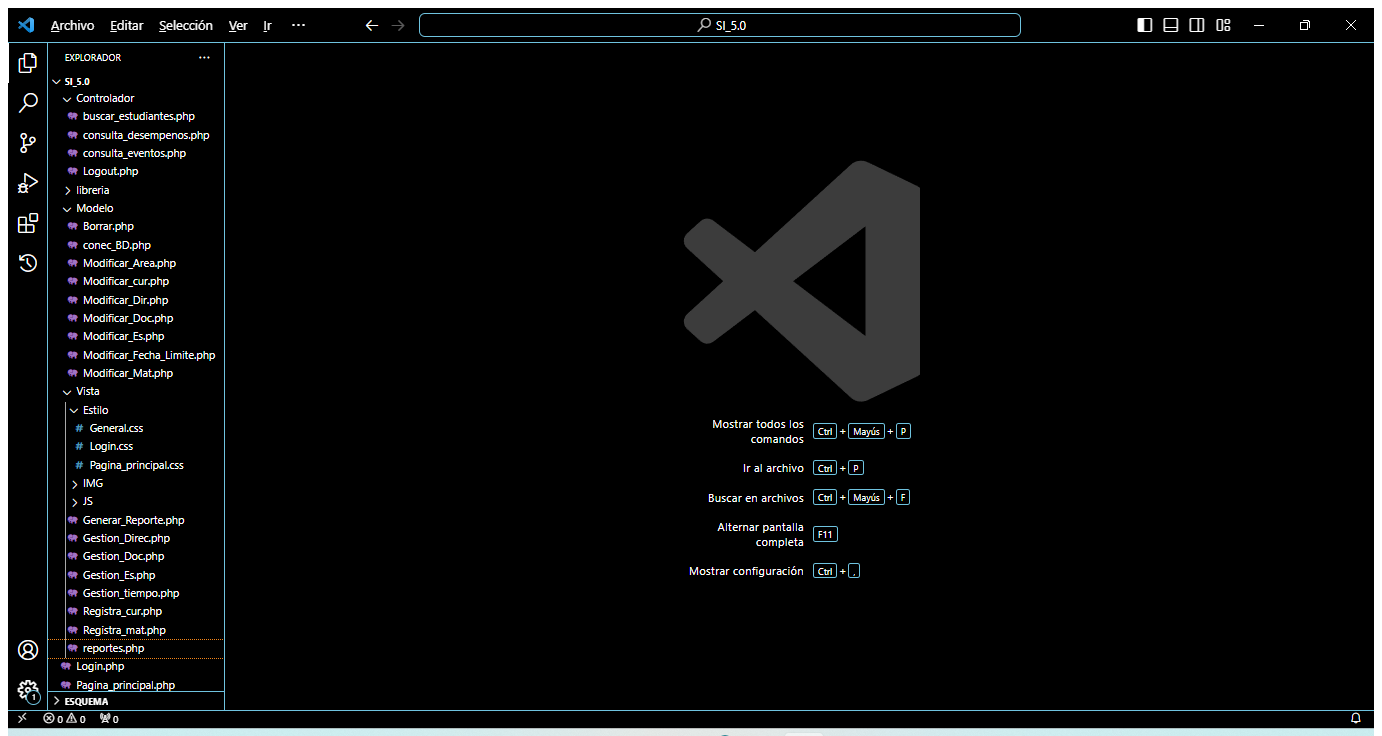
Un sistema de notas que permita gestionar de manera eficiente, eficaz la información académica de los estudiantes del Instituto Técnico Juan del Corral.

## Subsistema

A continuación, se describe la funcionalidad y las distintas pantallas que componen el subsistema de un sistema de notas de planillas de estudiantes:

* Pantalla de Inicio de Sesión (Login)
* Página principal(Registro de Notas)
* [Registrar Curso](https://docs.google.com/document/d/15ttF-1BAObXCRtErmcW6XXfLw0SEkwQv/edit#heading=h.dotp7yarg6iv)
* [Registrar Materias](https://docs.google.com/document/d/15ttF-1BAObXCRtErmcW6XXfLw0SEkwQv/edit#heading=h.td22sl7k9sbf)
* Gestión(usuarios) de Estudiantes , [Directivos](https://docs.google.com/document/d/15ttF-1BAObXCRtErmcW6XXfLw0SEkwQv/edit#heading=h.tj8whnhy6amr) , [Docentes](https://docs.google.com/document/d/15ttF-1BAObXCRtErmcW6XXfLw0SEkwQv/edit#heading=h.rfwx8jedt6tl).
* Gestión de tiempo
* Generar Reporte

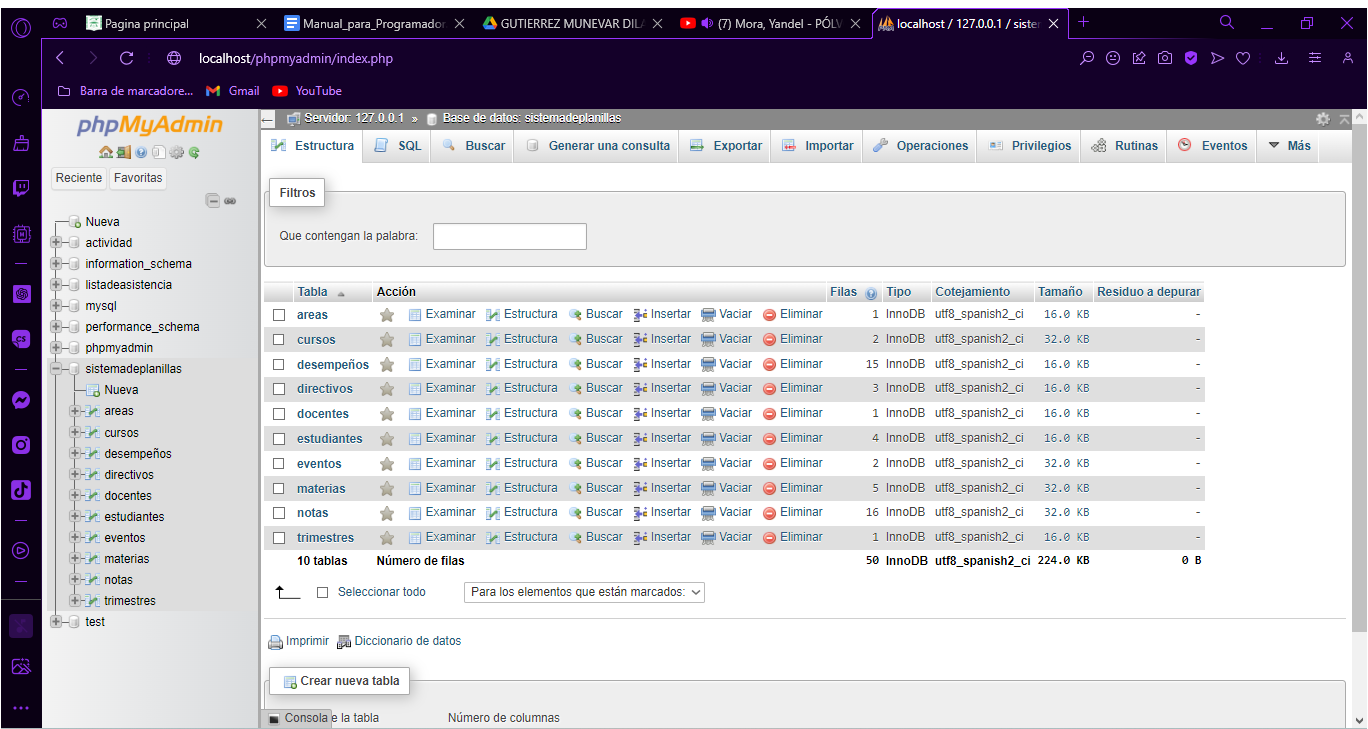
# Orden de Archivos y BD

Para la realización de el SI utilizamos la siguiente estructura de carpetas y archivos.

Inicialmente utilizamos un MVC adaptado a la manera en la que se le dio solución a la problemática del registro de notas .

En la carpeta principal encontramos 2 Archivos (Login.php y Pagina\_principal.php).

* Controlador : en este carpeta se encuentran los archivos PHP donde controlamos respuestas del servidor y muestra al cliente por medio de AJAX(buscar\_estudiantes.php, consulta\_eventos.php, consulta\_desempenos.php) y el archivo Logout.php donde controlamos el cierre de sesion de algún usuario
* librería: Utilizamos la librería DomPdf para poder generar reportes
* Modelo : en modelo manejamos acciones del CR**UD** de los diferentes módulos del sistema, además tenemos el archivo de conexión a la BD que trae la lógica de acceso a la BD
* Vista:En esta carpeta encontramos los archivos con las interfaces que se le van a mostrar al usuario además de 3 sub carpetas :
  + Estilo: Trae los archivos css de las interfaces
  + IMG:imagenes implementadas en las interfaces(Logos)
  + JS: Archivos Js implementados en el SI



La BD del SI esta conformada por las siguientes tablas: áreas, cursos, desempeños, directivos, docentes, estudiantes, eventos, materias , notas, trimestres; las cuales no están enlazadas por FK para mayor Simplicidad, Sin FK, el diseño de la base de datos y las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) pueden ser más simples y directas. No es necesario definir relaciones entre tablas ni preocuparse por el orden de inserción o eliminación de registros y también para mayor Flexibilidad, Sin restricciones de FK, tienes mayor flexibilidad para realizar cambios en la estructura de la base de datos, como la adición o eliminación de tablas o campos, sin preocuparte por la actualización de relaciones.

Nuestro SI trabaja con 3 Roles que son Administrador(Rector(a), Coordinador(a)), Usuario nivel 1(Docente) , Usuario nivel 2 (Estudiante).

El código principal de la conexión a la base de datos y el inicio de sesión, si desde el login no viene nada por la variable sesión lo redirige al Login. Si,si viene algo inicia las variables de sesion

# Login(Explicación código):

**<?php**

**// Comprueba si el parámetro 'documento' está configurado en la solicitud, y si no lo está, lo establece en null.**

**$documento = isset($\_REQUEST['documento']) ? $\_REQUEST['documento'] : null;**

**// Comprueba si el parámetro 'contrasena' está configurado en la solicitud, y si no lo está, lo establece en null.**

**$password = isset($\_REQUEST['contrasena']) ? $\_REQUEST['contrasena'] : null;**

**// Comprueba si el parámetro 'cargo' está configurado en la solicitud, y si no lo está, lo establece en null.**

**$Cargo = isset($\_REQUEST['cargo']) ? $\_REQUEST['cargo'] : null;**

**// Inicializa un arreglo para almacenar errores.**

**$errores = [];**

**// Comprueba si el método de solicitud HTTP es POST.**

**if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {**

**// Obtiene la entrada 'usuario' de los datos POST.**

**$input = $\_POST["usuario"];**

**// Comprueba si 'usuario' no está vacío y contiene un espacio.**

**if (!empty($input) && strpos($input, ' ') !== false) {**

**// Divide la entrada 'usuario' en 'nombre' y 'apellido'.**

**list($nombre, $apellido) = explode(" ", $input);**

**// Comprueba si 'nombre' y 'apellido' no están vacíos.**

**if (!empty($nombre) && !empty($apellido)) {**

**// Comprueba el valor de 'Cargo' y realiza acciones correspondientes.**

**if ($Cargo == "Rector(a)" || $Cargo == "Coordinador(a)") {**

**// Credenciales de conexión a la base de datos.**

**$usuario = "root";**

**$clave = '';**

**$miPDO = new PDO('mysql:host=localhost;dbname=sistemadeplanillas', $usuario, $clave);**

**// Prepara una consulta SQL para seleccionar datos de la tabla 'directivos'.**

**$miConsulta = $miPDO->prepare("SELECT Activo, contraseña FROM directivos WHERE Documento\_Direc=:documento AND Cargo=:cargo AND Nombre\_Directivo=:nombre AND Apellido\_Directivo=:apellido");**

**$miConsulta->execute([**

**'documento' => $documento,**

**'cargo' => $Cargo,**

**'nombre' => $nombre,**

**'apellido' => $apellido**

**]);**

**// Obtiene el resultado de la consulta a la base de datos.**

**$resultado = $miConsulta->fetch();**

**Comprueba si se encontró un registro coincidente.**

**if ($resultado) {**

**Comprueba si el usuario está activo.**

**if ((int)$resultado['Activo'] !== 1) {**

**$errores[] = 'Usuario inactivo. Por favor, contacte con el Administrador';**

**} else {**

**Verifica la contraseña utilizando password\_verify().**

**if (password\_verify($password, $resultado['contraseña'])) {**

**// Inicia una sesión y establece variables de sesión.**

**session\_start();**

**$\_SESSION["usuario"] = $nombre. " " . $apellido;**

**$\_SESSION['cargo'] = $Cargo;**

**$\_SESSION['documento'] = $documento;**

**// Establece una cookie con un nombre que indica que el mensaje ya se mostró.**

**setcookie("bienvenida\_mostrada", "true", time() + 5, "/"); // La cookie expirará en 5 segundos.**

**// Redirige al usuario a la página principal.**

**header('location: Pagina\_principal.php');**

**die();**

**} else {**

**$errores[] = 'Los datos ingresados son incorrectos. Por favor, verifica la información e inténtalo nuevamente';**

**}**

**}**

**} else {**

**$errores[] = 'Los datos ingresados son incorrectos. Por favor, verifica la información e inténtalo nuevamente';**

**}**

**}**

**Aca utilizamos elseif de si en el caso que el usuario sea Docente o Estudiantes consulte en la tabla estudiantes y docentes**

# Menú para diferentes usuarios

El menú cambiará dependiendo del usuario que ingrese al sistema .

# Variable de conexión, métodos y consulta utilizadas a lo largo del código:

<div class="select">

<select name="materia\_Registrar" id="materiaSelect">

<?php

$consulta= "SELECT \* FROM materias; ";

$result= $conex->query($consulta);

echo "<option selected disabled>Materia</option>";

while ($row = $result->fetch\_assoc()){

echo '<option value="' . $row['Nombre\_Materias'] . '" >' . $row['Nombre\_Materias'] . '</option>';

}

?>

</select>

</div>

fetch\_assoc(): Este método se utiliza para recuperar una fila de resultados de la consulta y devuelve un arreglo asociativo (array) que representa esa fila.

$conex: Parece ser una variable que almacena una conexión a la base de datos. Antes de este código, es probable que hayas establecido la conexión a la base de datos utilizando funciones como mysqli\_connect() o similar.

query(): ejecuta la consulta de la BD

**<!-- En caso de ser Estudiante o Admin Académico muestra el nombre del profesor que dicta la materia -->**

**<?php**

**if($Cargo == "Estudiante" || $Cargo == "Rector(a)" || $Cargo == "Coordinador(a)" ){**

**if(isset($materiaId)){**

**// Realiza una consulta para obtener el nombre del docente asignado a esta materia**

**$consultaDocente = $miPDO->prepare("SELECT Nombre\_Docente, Nombre\_Docente\_2, Apellido\_Docente, Apellido\_Docente\_2 FROM docentes WHERE Nombre\_Materia\_Asignada LIKE :materiaNombre");**

**$consultaDocente->bindValue(':materiaNombre', '%"'.$materiaId.'"%', PDO::PARAM\_STR);**

**$consultaDocente->execute();**

**if ($consultaDocente->rowCount() > 0) {**

**$docenteInfo = $consultaDocente->fetch();**

**$primerNombre = $docenteInfo['Nombre\_Docente'];**

**$segundoNombre = $docenteInfo['Nombre\_Docente\_2'];**

**$primerApellido = $docenteInfo['Apellido\_Docente'];**

**$segundoApellido = $docenteInfo['Apellido\_Docente\_2'];**

**// Puedes usar estas variables en tu HTML para mostrar el nombre del docente asignado a esta materia**

**echo '<h4 id="nomDocente">Docente: '.$primerNombre.' '.$segundoNombre.' '.$primerApellido.' '.$segundoApellido.'</h4>';**

**}**

**}**

**}**

**?>**

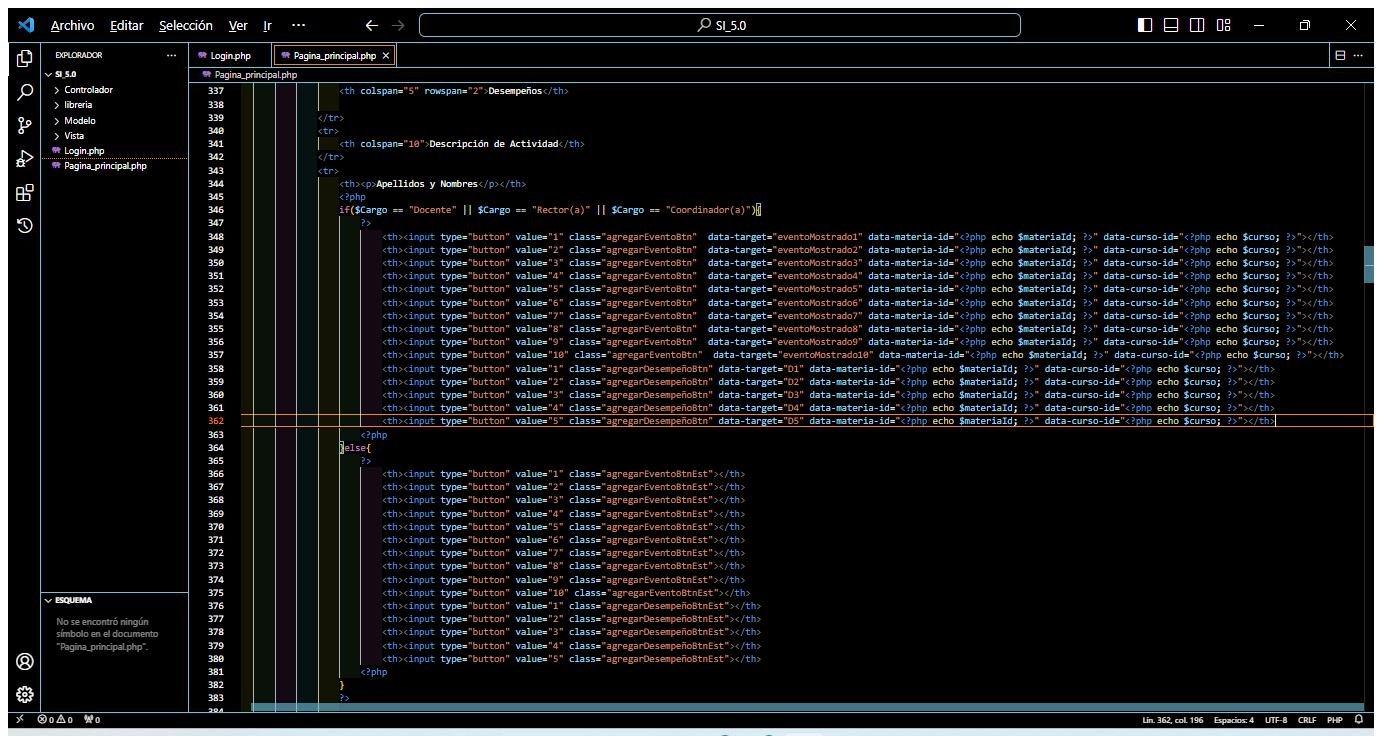
fetch(): Este método se utiliza para obtener la siguiente fila de resultados de la consulta.

rowCount(): Este método se utiliza para obtener el número de filas afectadas por la consulta SQL.

execute(): Este método se utiliza para ejecutar la consulta SQL preparada con los valores enlazados. Realiza la consulta en la base de datos.

Se utiliza el tipo de parámetro PDO::PARAM\_STR para indicar que se trata de una cadena.

bindValue(':materiaNombre', '%"'.$materiaId.'"%', PDO::PARAM\_STR)**: Este método se utiliza para enlazar un valor a un marcador de posición en la consulta SQL preparada**



**El atributo** data-target **en HTML se utiliza para especificar un destino o un selector que se relaciona con un elemento específico. Este atributo es comúnmente utilizado para vincular un elemento HTML (como un botón) con un elemento objetivo en una página web, a menudo con fines de interacción en JavaScript o para definir relaciones en CSS.**

**A lo largo del código utilizamos los siguientes Métodos**

fetch(PDO::FETCH\_ASSOC)**:** fetch() **es un método utilizado para obtener una fila de resultados de una consulta SQL.** PDO::FETCH\_ASSOC **se utiliza como argumento para** fetch() **para que los resultados se devuelvan en un array asociativo, donde los nombres de las columnas de la base de datos se utilizan como claves.**

* bindParam**:** bindParam **es un método de PDO que se utiliza para enlazar valores a marcadores de posición en una consulta SQL. Se utiliza para asignar valores a los marcadores antes de ejecutar la consulta.**
* json\_decode**: Esta función de PHP se utiliza para decodificar una cadena JSON y convertirla en una estructura de datos en PHP, generalmente en forma de un array o un objeto.**
* array\_sum**:** array\_sum **es una función de PHP que se utiliza para calcular la suma de los valores en un array numérico.**
* empty**:** empty **es una función que se utiliza para verificar si una variable está vacía o si su valor es falso. Devuelve** true **si la variable está vacía o** false**, y** false **si la variable contiene algún valor.**
* count**:** count **es una función de PHP que se utiliza para contar el número de elementos en un array. Puede ser útil para determinar la cantidad de elementos en un array.**
* floatval**:** floatval **es una función que se utiliza para convertir una variable en un valor de tipo float (número decimal de punto flotante).**
* array\_filter**:** array\_filter **es una función que se utiliza para filtrar los elementos de un array basándose en una función de devolución de llamada que determina si un elemento debe mantenerse en el array o no.**
* number\_format**:** number\_format **es una función que se utiliza para formatear un número con miles y decimales. Puedes especificar el número de decimales, el separador de miles y el separador decimal.**
* json\_encode**:** json\_encode **es la contraparte de** json\_decode**. Se utiliza para convertir una estructura de datos en PHP (como un array o un objeto) en una cadena JSON.**

* **La verificación** if(isset($\_POST["regis-est"])) **verifica si se ha enviado un formulario con un campo llamado "regis-est". Esto es típico de un formulario HTML donde se usa un botón para enviar los datos del estudiante para su registro.**
* **Se recopila el valor del campo** Documento\_estudiante **del formulario, que se envía mediante** $\_REQUEST['Documento\_estudiante'] **y se almacena en la variable** $doc**.**
* **Se construye una consulta SQL para buscar si ya existe un estudiante con el mismo número de documento en la tabla "estudiantes". La consulta se realiza en la base de datos usando la conexión definida en** $conex**.**
* **La variable** $doble **almacena el resultado de la consulta SQL. Si** $doble **tiene filas (es decir,** $doble->num\_rows > 0**), significa que ya existe un estudiante con el mismo número de documento en la base de datos.**
* **Si no hay estudiantes duplicados (es decir,** $doble->num\_rows **es 0), se recopilan los datos del estudiante del formulario, como el nombre, el apellido, la contraseña, el número de curso al que pertenece, etc.**
* **Se prepara una consulta SQL para insertar un nuevo estudiante en la tabla "estudiantes" utilizando** PDO::prepare**.**
* **Luego, se ejecuta la consulta con** $miInsert->execute() **y se pasan los valores correspondientes como un array asociativo.**
* **Después de ejecutar la consulta de inserción, se verifica si se realizó con éxito (**if($miInsert)**). Si se realiza con éxito, se muestra un mensaje indicando que el estudiante se ha registrado correctamente.**
* **Si ya existe un estudiante con el mismo número de documento, se muestra un mensaje de error indicando que ya hay un estudiante registrado con ese documento.**

password\_hash:  **es una función de PHP que se utiliza para almacenar contraseñas de forma segura en una base de datos mediante el uso de técnicas de hashing seguras, proporcionando una mayor seguridad a los usuarios.(Encriptar)**

**Los módulos del sistema (Gestion Estudiantes, Directivos, Docentes) utiliza esta misma estructura solo que cambia dependiendo de la tabla de la BD**

* **La primera línea** if(isset($\_POST["consul-est"])) **verifica si se ha enviado un formulario con un campo llamado "consul-est". Esto sugiere que el código se ejecuta cuando se envía un formulario con la intención de buscar estudiantes en la base de datos.**
* **Luego, se verifica si se ha proporcionado un valor para el campo "Nombre\_estudiante" en el formulario. Esto se hace mediante** if(strlen($\_POST["Nombre\_estudiante"]) >= 1)**, lo que significa que el código solo continuará si se ha ingresado al menos un carácter en el campo de búsqueda de nombre de estudiante.**
* **Si se ha proporcionado un nombre de estudiante válido, se recopila ese valor en la variable** $nombre**.**
* **A continuación, se prepara una consulta SQL utilizando PDO para buscar estudiantes en la tabla "estudiantes" cuyo nombre coincida con el valor proporcionado. La consulta se ejecuta utilizando** $miConsulta->execute()**.**
* **Se verifica si la consulta ha encontrado algún estudiante en la base de datos utilizando** $miConsulta->rowCount()**. Si la consulta no ha devuelto resultados (es decir,** $miConsulta->rowCount() **es 0), se muestra un mensaje indicando que no se encontraron estudiantes con ese nombre.**
* **Si la consulta ha encontrado estudiantes, se muestra una tabla que contiene los detalles de los estudiantes que coinciden con el nombre de búsqueda. Los detalles incluyen el nombre, apellidos, número de documento y número de curso al que pertenecen.**
* **Para cada estudiante encontrado, se crea una fila en la tabla HTML utilizando un bucle** foreach**. En cada fila se muestran los detalles del estudiante y se proporcionan enlaces para modificar el estudiante o habilitar/inhabilitar su cuenta, dependiendo del estado de activación del estudiante.**

**Para ejecutar las búsquedas , no solo puede ser por medio del nombre , si no que también por otros campos, de igual manera se repite en varios modulos   
  
La primera línea** if (isset($\_GET["Id\_estudiantes"])) **verifica si se ha recibido un parámetro llamado "Id\_estudiantes" a través de la URL. Esto sugiere que el código se ejecuta cuando se accede a una página o se hace clic en un enlace que proporciona el** Id\_estudiantes **como parte de la URL.**

* **Luego, se recopila el valor del parámetro** Id\_estudiantes **de la URL utilizando** $\_REQUEST['Id\_estudiantes'] **y se almacena en la variable** $codigoEst**.**
* **Se prepara una consulta SQL utilizando PDO para actualizar el estado de activación del estudiante. La consulta se estructura de manera que si el estudiante está activo (con un valor de** Activo **igual a 1), se desactiva (se establece a 0), y viceversa. Esto se logra utilizando la expresión** IF(Activo = 1, 0, 1) **en la consulta.**
* **Luego, se ejecuta la consulta utilizando** $miUpdateEst->execute() **y se pasa el valor del parámetro** Id\_estudiantes **como un marcador de posición llamado** :Id\_estudiantes**.**
* **Después de ejecutar la consulta de actualización, se verifica si se realizó con éxito (**if ($miUpdateEst)**). Si la actualización se realiza con éxito, se redirige al usuario a la página "Gestion\_Es.php" utilizando** header("location:../Vista/Gestion\_Es.php")**.**

**Este código lo utilizamos para inhabilitar usuarios**

# Trigger

Los trigger del SI son :

**AFTER\_DELETE CURSO**

CREATE TRIGGER `After\_Delete\_Curso` AFTER DELETE ON `cursos`

FOR EACH ROW BEGIN

DECLARE curso\_numero INT;

-- Obtener el número del curso eliminado

SET curso\_numero = OLD.Numero\_Curso;

-- Actualizar estudiantes

UPDATE estudiantes

SET Numero\_Curso\_pertenece = 0

WHERE Numero\_Curso\_pertenece = curso\_numero;

-- Eliminar el número de curso del arreglo en la tabla areas

UPDATE areas

SET Numero\_Curso = JSON\_REMOVE(Numero\_Curso, JSON\_UNQUOTE(JSON\_SEARCH(Numero\_Curso, 'one', curso\_numero)))

WHERE JSON\_SEARCH(Numero\_Curso, 'one', curso\_numero) IS NOT NULL;

-- Eliminar el número de curso del arreglo en la tabla docentes

UPDATE docentes

SET Numero\_curso\_asignado = JSON\_REMOVE(Numero\_curso\_asignado, JSON\_UNQUOTE(JSON\_SEARCH(Numero\_curso\_asignado, 'one', curso\_numero)))

WHERE JSON\_SEARCH(Numero\_curso\_asignado, 'one', curso\_numero) IS NOT NULL;

END

**AFTER\_DELETE\_MATERIA**

CREATE TRIGGER `After\_Delete\_Materia` AFTER DELETE ON `materias`

FOR EACH ROW BEGIN

DECLARE materia\_name VARCHAR(255);

-- Obtener el nombre de la materia eliminada

SET materia\_name = OLD.Nombre\_Materias;

-- Eliminar el nombre de la materia del arreglo en la tabla areas

UPDATE areas

SET Nom\_mat\_pertenecen = JSON\_REMOVE(Nom\_mat\_pertenecen, JSON\_UNQUOTE(JSON\_SEARCH(Nom\_mat\_pertenecen, 'one', materia\_name)))

WHERE JSON\_SEARCH(Nom\_mat\_pertenecen, 'one', materia\_name) IS NOT NULL;

-- Eliminar el nombre de la materia del arreglo en la tabla docentes

UPDATE docentes

SET Nombre\_Materia\_Asignada = JSON\_REMOVE(Nombre\_Materia\_Asignada, JSON\_UNQUOTE(JSON\_SEARCH(Nombre\_Materia\_Asignada, 'one', materia\_name)))

WHERE JSON\_SEARCH(Nombre\_Materia\_Asignada, 'one', materia\_name) IS NOT NULL;

END

**AFTER\_UPDATE\_MATERIA**

CREATE TRIGGER `after\_update\_Materia` AFTER UPDATE ON `materias`

FOR EACH ROW BEGIN

-- Actualizar el arreglo Nom\_mat\_pertenecen en la tabla areas

UPDATE areas

SET Nom\_mat\_pertenecen = JSON\_REPLACE(Nom\_mat\_pertenecen, JSON\_UNQUOTE(JSON\_SEARCH(Nom\_mat\_pertenecen, 'one', OLD.Nombre\_Materias)), NEW.Nombre\_Materias)

WHERE JSON\_SEARCH(Nom\_mat\_pertenecen, 'one', OLD.Nombre\_Materias) IS NOT NULL;

-- Actualizar el arreglo Nombre\_Materia\_Asignada en la tabla docentes

UPDATE docentes

SET Nombre\_Materia\_Asignada = JSON\_REPLACE(Nombre\_Materia\_Asignada, JSON\_UNQUOTE(JSON\_SEARCH(Nombre\_Materia\_Asignada, 'one', OLD.Nombre\_Materias)), NEW.Nombre\_Materias)

WHERE JSON\_SEARCH(Nombre\_Materia\_Asignada, 'one', OLD.Nombre\_Materias) IS NOT NULL;

-- Actualizar el campo Curso en la tabla notas

UPDATE notas

SET Nom\_mat = NEW.Nombre\_Materias

WHERE Nom\_mat = OLD.Nombre\_Materias;

-- Actualizar el campo Curso en la tabla eventos

UPDATE eventos

SET Nom\_mat = NEW.Nombre\_Materias

WHERE Nom\_mat = OLD.Nombre\_Materias;

-- Actualizar el campo Curso en la tabla desempeños

UPDATE desempeños

SET Nom\_mat = NEW.Nombre\_Materias

WHERE Nom\_mat = OLD.Nombre\_Materias;

END

**AFTER\_UPDATE\_CURSO**

CREATE TRIGGER `after\_update\_curso` AFTER UPDATE ON `cursos`

FOR EACH ROW BEGIN

-- Actualizar estudiantes

UPDATE estudiantes

SET Numero\_Curso\_pertenece = NEW.Numero\_Curso

WHERE Numero\_Curso\_pertenece = OLD.Numero\_Curso;

-- Actualizar el arreglo Numero\_Curso\_asignado en la tabla docentes

UPDATE docentes

SET Numero\_curso\_asignado = JSON\_REPLACE(Numero\_curso\_asignado, JSON\_UNQUOTE(JSON\_SEARCH(Numero\_curso\_asignado, 'one', OLD.Numero\_Curso)), NEW.Numero\_Curso)

WHERE JSON\_SEARCH(Numero\_curso\_asignado, 'one', OLD.Numero\_Curso) IS NOT NULL;

-- Actualizar el arreglo Numero\_Curso en la tabla areas

UPDATE areas

SET Numero\_Curso = JSON\_REPLACE(Numero\_Curso, JSON\_UNQUOTE(JSON\_SEARCH(Numero\_Curso, 'one', OLD.Numero\_Curso)), NEW.Numero\_Curso)

WHERE JSON\_SEARCH(Numero\_Curso, 'one', OLD.Numero\_Curso) IS NOT NULL;

-- Actualizar el campo Num\_cur en la tabla notas

UPDATE notas

SET Num\_cur = NEW.Numero\_Curso

WHERE Num\_cur = OLD.Numero\_Curso;

-- Actualizar el campo Num\_cur en la tabla eventos

UPDATE eventos

SET Num\_cur = NEW.Numero\_Curso

WHERE Num\_cur = OLD.Numero\_Curso;

-- Actualizar el campo Num\_cur en la tabla desempeños

UPDATE desempeños

SET Num\_cur = NEW.Numero\_Curso

WHERE Num\_cur = OLD.Numero\_Curso;

END

**BEFORE INSERT DESEMPEÑOS**

CREATE TRIGGER `before\_insert\_desempeños` BEFORE INSERT ON `desempeños`

FOR EACH ROW BEGIN

DECLARE trimestre\_nombre VARCHAR(255);

DECLARE trimestre\_ano INT;

-- Obtén el nombre del trimestre que se va a insertar en la tabla desempeños

SET trimestre\_nombre = NEW.trimestre;

-- Busca el año correspondiente en la tabla trimestres

SELECT YEAR(fecha\_inicio) INTO trimestre\_ano FROM trimestres WHERE nombre\_trimestre = trimestre\_nombre;

-- Inserta el año en la columna correspondiente en la tabla desempeños

SET NEW.Ano = trimestre\_ano;

END

**BEFORE\_INSERT\_EVENTOS**

CREATE TRIGGER `before\_insert\_eventos` BEFORE INSERT ON `eventos`

FOR EACH ROW BEGIN

DECLARE trimestre\_nombre VARCHAR(255);

DECLARE trimestre\_ano INT;

-- Obtén el nombre del trimestre que se va a insertar en la tabla eventos

SET trimestre\_nombre = NEW.trimestre;

-- Busca el año correspondiente en la tabla trimestres

SELECT YEAR(fecha\_inicio) INTO trimestre\_ano FROM trimestres WHERE nombre\_trimestre = trimestre\_nombre;

-- Inserta el año en la columna correspondiente en la tabla eventos

SET NEW.Ano = trimestre\_ano;

END

**BEFORE\_INSERT\_NOTAS**

CREATE TRIGGER `before\_insert\_notas` BEFORE INSERT ON `notas`

FOR EACH ROW BEGIN

DECLARE trimestre\_nombre VARCHAR(255);

DECLARE trimestre\_ano INT;

-- Obtén el nombre del trimestre que se va a insertar en la tabla notas

SET trimestre\_nombre = NEW.trimestre;

-- Busca el año correspondiente en la tabla trimestres

SELECT YEAR(Fecha\_inicio) INTO trimestre\_ano FROM trimestres WHERE nombre\_trimestre = trimestre\_nombre;

-- Inserta el año en la columna correspondiente en la tabla notas

SET NEW.Ano = trimestre\_ano;

END

# Glosario

Este punto contendrá la definición de todos los términos utilizados, y se considerará de interés para la comprensión del sistema.

| **Término** | **Descripción** |
| --- | --- |
| Registro de Calificaciones | Es en donde se registran las calificaciones de los estudiantes para un conjunto específico de cursos o asignaturas. |
| BD(Base de Datos) | Es un sistema de almacenamiento de datos que contiene información sobre los estudiantes, como nombres, identificaciones, calificaciones y asistencia. |
| Pantalla de Inicio de Sesión | Permite a los usuarios iniciar sesión con sus credenciales y acceso al sistema a través de nombres de usuario y contraseñas. |
| Sistema de Gestión de Planillas de Notas (SGPN) | El software permite a los usuarios administrar y gestionar las calificaciones y notas de los estudiantes. |
| Rol de Usuario | Un conjunto de permisos y funciones asignados a un usuario que determina su nivel de acceso y las acciones que puede realizar en el sistema. |