

MANUAL DEL SISTEMA AUTOMATIZADO DE NOTAS (SAN)

Realizado por:

Grethmar Rojas

Jean Mora

Álvaro de Arriba

Daniel Sulbaran

Diego Zambrano

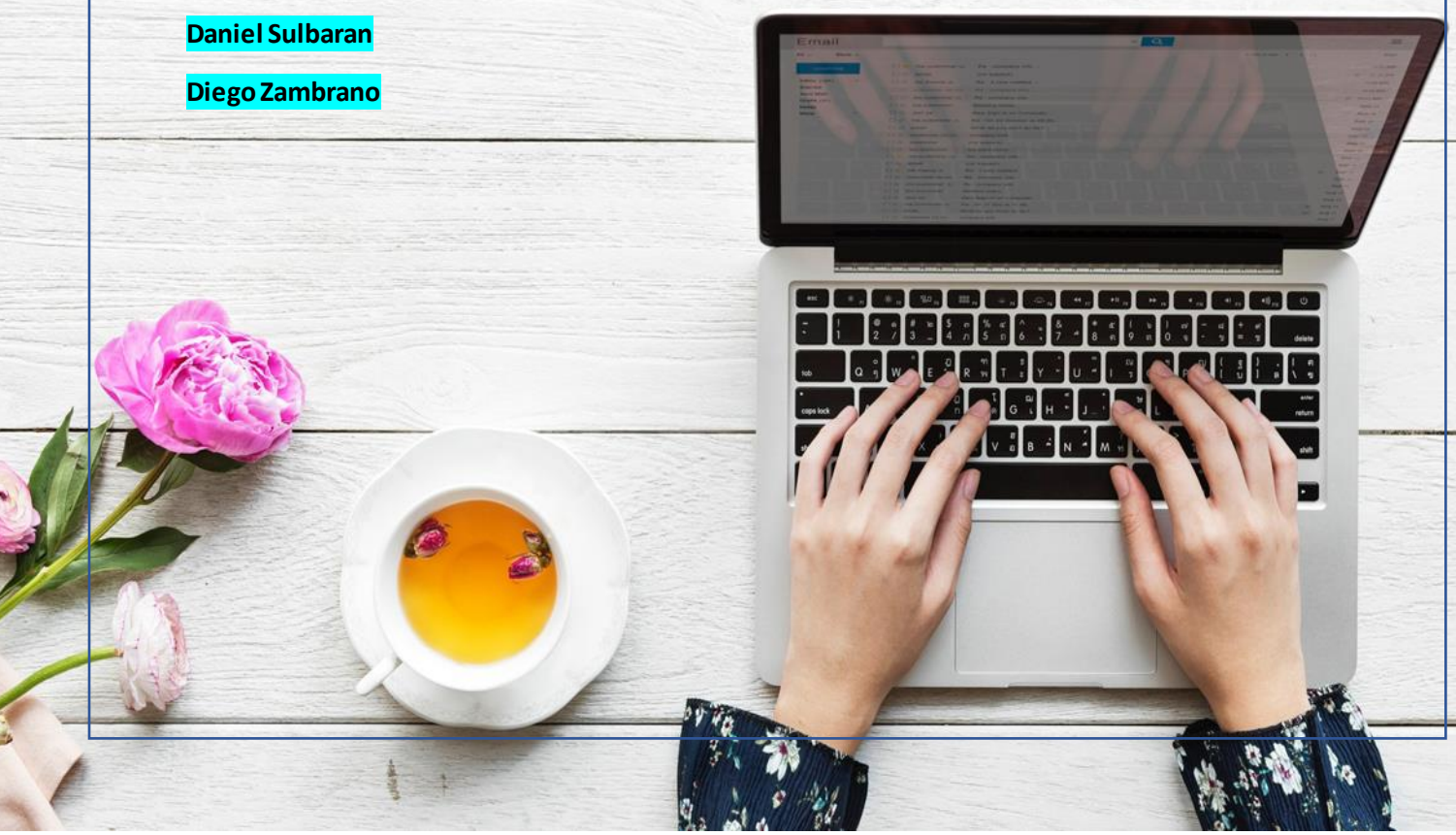


TABLA DE CONTENIDO

Introducción.....	3
Arquitectura del SAN	4
Arquitectura del SAN	5
Registro e inicio de sesión	5
Funcionalidades del sistema SAN.....	7
Gestión de usuarios	7
Plantilla estudiantil	8
Carga de notas.....	9
Generar reportes	9
Conclusión	10



INTRODUCCIÓN

El Sistema Automatizado de Notas (SAN) es una aplicación web que permite a los usuarios registrar, consultar y gestionar notas de estudiantes. El sistema cuenta con diferentes funcionalidades para llevar a cabo estas tareas, como la gestión de los usuarios registrados en el sistema, registro y consulta de la plantilla estudiantil, carga de notas según grado y sección, generación de reportes como constancias, entre otros.

El presente manual tiene como objetivo proporcionar al usuario una guía detallada del sistema, explicando las diferentes funcionalidades y los procesos involucrados en su uso. El manual está diseñado para ser utilizado por los programadores que necesiten realizar cambios o mejoras en el sistema.

ARQUITECTURA DEL SAN

El sistema se divide en diferentes componentes o capas, cada una con una función específica. Principalmente se divide en el sistema de login y sistema de notas, cada uno con sus capas:

- Capa de presentación: esta capa se encarga de la interfaz de usuario. Aquí se definen las vistas y plantillas que el usuario ve y con las que interactúa.
- Capa de controladores: esta capa se encarga de procesar las solicitudes del usuario, realizando las operaciones correspondientes y devolviendo una respuesta al usuario.
- Capa de modelos: esta capa se encarga de interactuar con la base de datos y proporciona los métodos necesarios para realizar las operaciones de lectura, escritura y eliminación de datos.
- Base de datos: esta capa almacena los datos del sistema. En este caso, se utiliza una base de datos relacional MySQL.

Imagen 1. Podemos observar la carpeta principal SAN3.2, donde se tiene las diferentes capas de modelo, vista, controlador y recursos del sistema Login.

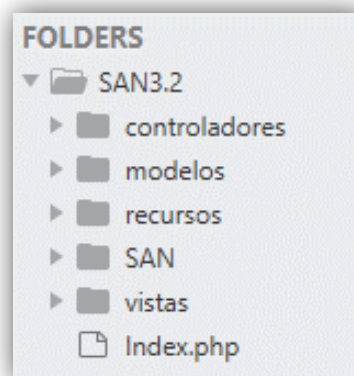
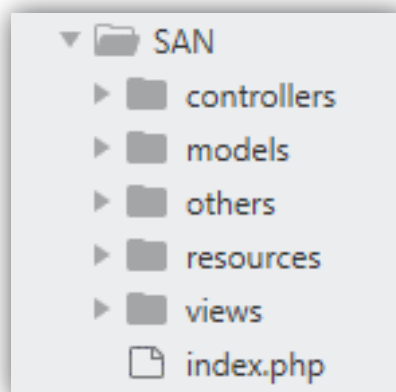


Imagen 2. La carpeta llamada SAN desglosa toda la división y el patrón de arquitectura MVC del sistema automatizado de notas



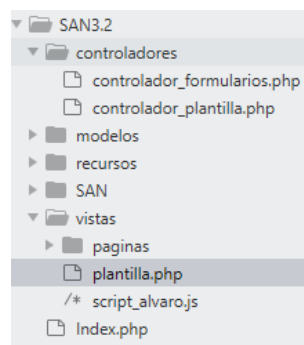
FUNCIONALIDADES DEL SISTEMA LOGIN

❖ Registro e ingreso al sistema

En el sistema del login, existe la opción de registrar un nuevo usuario e iniciar sesión con un usuario ya registrado.

The image displays two user interface mockups for a login system. The left mockup, titled "Registro", features a vertical form with four input fields: "User:" (with a person icon), "Email:" (with an envelope icon), "Password:" (with a lock icon), and "Confirmar Password:" (with a lock icon). Below these fields are two buttons: "Registrarse" and "Logearse". The right mockup, titled "Sistema Automatizado de Notas", features a vertical form with two input fields: "Email:" (with an envelope icon) and "Password:" (with a lock icon). Below these fields are two buttons: "Ingresar" and "Registrarse".

A nivel de programación, se creó la vista `index.php` el cual instancia la clase `ControladorPlantilla()` en el objeto `$plantilla` quien llama al método `ctrTraerPlantilla()`. De esta manera, en la carpeta `controladores`, existe el archivo `controlador_plantilla.php`, donde están declaradas dicha clase y función no estática el incluye (include) el archivo `plantilla.php` según su ubicación en la carpeta `vistas`.



En la carpeta de las vistas, está la carpeta paginas donde se desarrollaron cada una de las vistas del sistema login: Registro, ingreso y error404. En las vistas de registro e ingreso crearon los respectivos formularios de método POST con cada uno de los input solicitando la información requerida, si el usuario ejecuta una acción de la vista Registro, se llama instancia de la clase ControladorFormularios la función estática ctrRegistro() los cuales están declarados en el archivo dentro de los controladores controlador_formularios.php.

Cabe resaltar, que primero es necesario la conexión a la base de datos *Revisar en la carpeta modelos el archivo conexion.php*

El controlador del registro pregunta si se obtuvo la variable \$_POST["RegistroUser"] del formulario, en caso de ser positivo, declara la variable \$tabla con el nombre de la base de datos, en este caso "perso" y la variable array \$datos con los valores a almacenar User, email y password. Dichos datos se pasan al modelo instanciándolo en la variable \$respuesta.

```
class ControladorFormularios{

    #registro
    static public function ctrRegistro(){

        if(isset($_POST["RegistroUser"])){          #nos aseguramos que si se haya guardado la
            variable

            $tabla = "perso"; #nombre de nuestra tabla en la bd

            $datos = array("user" => $_POST["RegistroUser"], #creamos un array con los datos
                que vamos a guardar
                "email" => $_POST["RegistroEmail"], #En el campo Email se pondra
                lo que se coloque en el RegistroEmail
                "password" => $_POST["RegistroPassword"]);

            #Le pasamos los datos al modelo haciendo una instancia con la variable de
            respuesta
            #Y le pasamos lo que nos pida (la tabla y los datos)
            $respuesta = modeloformulario::mdlregistro($tabla, $datos);

            return $respuesta;
        }
    }
}
```

En el archivo formulario.modelo existe la clase modeloformulario y la función estática mdlregistro con las variables declaradas en el controlador \$tabla y \$datos. De esta manera, se verifica primero si existe ese usuario a registrar, si es diferente a existe, entonces se realiza la sentencia SQL: INSERT INTO almacenando cada dato en la tabla. Con BindParam se desocultan los valores en la sentencia SQL, si ejecuta se retorna OK.

Para el ingreso, tiene un proceso parecido, donde el controlador ctrIngreso recibe la variable POST isset(\$_POST["IngresoEmail"]) del formulario de la vista de ingreso y declara la variable \$tabla con la base de datos "perso", \$item con el email el cual es la columna para buscar coincidencia y \$valor con la variable POST obtenida del formulario el cual debe coincidir para ingresar. Y así se instancia con la función del modelo mdlSeleccionarRegistros. Si la respuesta del email obtenida del modelo es igual a la variable POST obtenida del formulario y la respuesta de la contraseña obtenida del modelo es igual a la variable POST (contraseña) obtenida del formulario coinciden, entonces, la variable de sesión será ok y permite el ingreso, sino muestra un error de credenciales invalidas.

En el caso del modelo (ingreso), la función estática mdlSeleccionarRegistros con los valores declarados en el controlador ctrIngreso \$tabla, \$item y \$valor, se conecta con la base de datos perso, y selecciona todos los registros donde la variable item coincida, devolviendo con el fetch().

FUNCIONALIDADES DEL SAN

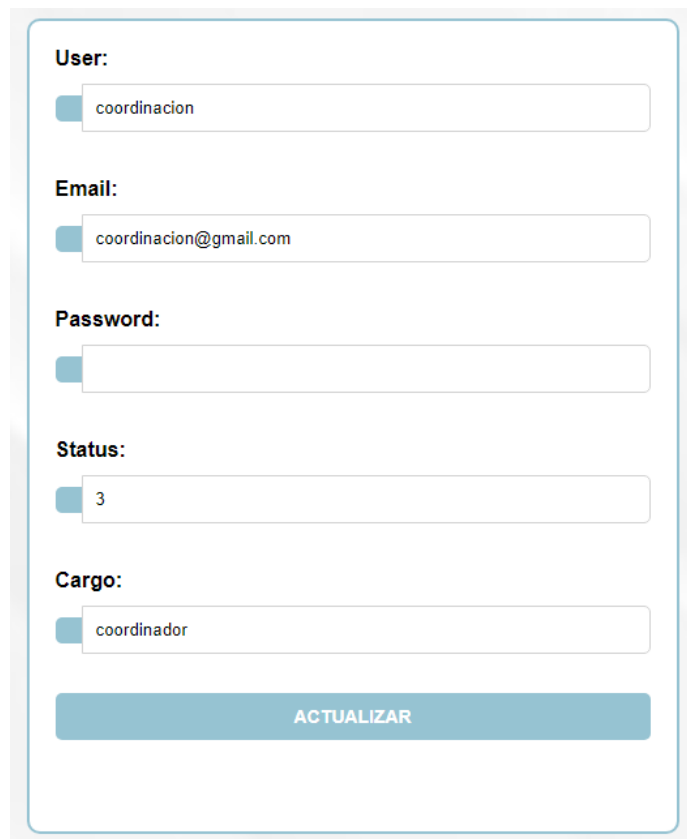
❖ Gestión de usuarios

Al ingresar al sistema automatizado de notas, en el menú desplegable ubicado a la izquierda, se encuentra la opción Gestión de usuarios. Donde se visualiza una tabla con los registros de los usuarios del sistema.

&&	USUARIO	PASSWORD	EMAIL	ESTATUS	CARGO	FECHA	
1	grethmar	123	rojas@gmail.com	4	admin	00/00/00	EDITAR
2	Prueba	12345	prueba@gmail.com	0	nuevo	12/03/23	EDITAR
3	coordinacion	123	coordinacion@gmail.com	3	coordinador	18/03/23	EDITAR

La información se muestra en la vista usuarios.php (Recordar que ya esta parte es el SAN, ubicar la carpeta SAN donde tiene las diferentes carpetas del patron MVC), para mostrar la información se instancia con el ctrSeleccionarRegistros de la clase ControladorFormularios ubicado en la carpeta controllers del SAN. En dicha función se declaran las variables \$tabla y se instancia en la variable respuesta el modelo mdlSeleccionarRegistros. En dicho modelo (ubicación del archivo: SAN/models/MOD_form.php) se ubican los registros por coincidencia donde el valor \$item y \$valor son nulos, por lo que se ejecuta una secuencia SQL seleccionando todos los registros de la base de datos.

Para editar la información se direcciona a la vista editar con el valor del ID del usuario a modificar, en dicha vista editar.php, primero se valida que se haya obtenido la variable POST del id mencionada anteriormente, y así se declaran las variables \$item (columna id) y \$valor (valor del id obtenido del POST), instancia el método ctrSeleccionarRegistros el cual llama del modelo mdlSeleccionarRegistros, la diferencia es que en esta acción el valor de \$item y \$valor no son nulos, por lo que, ejecuta una secuencia SQL de SELECT pero con coincidencia con el item, retornando dicha coincidencia (fetch). Y así poder mostrar en la vista editar.php, los datos del usuario en específico a editar, es un formulario el cual guarda los nuevos valores y al enviarlo, se instancia el método no estático ctrActualizarRegistros, donde se declaran las variables \$tabla y \$datos (tipo array, almacenando cada uno de los nuevos datos) y la respuesta la solicita del mdlActualizarRegistros el cual con la secuencia SQL UPDATE, actualiza la información en la base de datos, se enlazan parámetros y se desocultan los valores, si ejecuta correctamente devuelve OK.

Un formulario web para actualizar los datos de un usuario. El formulario está encerrado en un recuadro con una sombra y contiene los siguientes campos: 'User:' con el valor 'coordinacion', 'Email:' con el valor 'coordinacion@gmail.com', 'Password:' (vacío), 'Status:' con el valor '3', y 'Cargo:' con el valor 'coordinador'. Cada campo tiene un icono de ojo para alternar la visibilidad. Al final del formulario hay un botón azul con el texto 'ACTUALIZAR'.

❖ Plantilla estudiantil: Cargar y consultar(ver).

En la siguiente opción del menú lateral, está la plantilla estudiantil donde se pueden cargar (registrar un estudiante a la plantilla) y ver (consultar la plantilla). Para cargar un estudiante, se visualiza el formulario de la vista `c_plantilla.php`, al enviar el formulario se instancia el método `ctrRegistro` el cual es similar a al controlador de registro del login, con la diferencia de la base de datos y los datos, solicitando respuesta al modelo (`mdlRegistro`).

Para consultarla plantilla estudiantil, se requiere del controlador el método `ctrSeleccionarAula`, para así hacer la consulta en el modelo `mdlSeleccionarAula` según el grado y la sección seleccionada por el usuario. Se puede editar y desincorporar, para la modificación de datos se redirecciona a la vista `editarestu.php` el cual muestra la información del estudiante mediante el controlador `ctrSeleccionarEstu` quien le solicita respuesta al modelo `mdlSeleccionarEstu`, con un formulario permite modificar los datos del estudiante. El proceso es similar a la actualización de los registros de gestión de usuario mencionado anteriormente, con la diferencia de que esta conectado es a la tabla "estu". Controlador `ctrActualizarEstu` y el modelo es `mdlActualizarEstu`.

Otra opción de la tabla de la plantilla estudiantil es desincorporar estudiante, mediante la instanciación de `ctrDesincorporarEstu`, el cual valida si se obtuvo la variable `POST DesincEstu` de la vista, declara la variable `$id` (valor ID del estudiante seleccionado) y `%motivo` (variable `POST` motivo, el cual se obtiene del formulario -vista- donde se solicita el motivo del egreso). Así solicita respuesta al modelo `mdlEgresarEstudiante` (método no estático) el cual mediante secuencias SQL selecciona el estudiante de la tabla `estu` (coincidente con el `id`), inserta los datos de dicho estudiante en la tabla `egresos` (`INSERT INTO`) y lo elimina de la tabla `estu` (`DELETE FROM`).

También permite ver la plantilla estudiantil (EGRESOS), mediante el controlador `ctrSeleccionarEgresos` quien solicita respuesta al modelo `mdlSeleccionarEgresos`, donde ejecuta la consulta `SLQ SELECT`, todos los registros de la tabla `egresos`.

Registre un estudiante

ID:

Nombre:

Apellido:

Genero:

Selecione:

Grado:

Selecione:

Seccion:

Selecione:

Representante:

REGISTRARSE

Selecione el Grado

1

Selecciona una sección:

☐ Sección A
☐ Sección B
☐ Sección C

CONTINUAR

VER EGRESOS

CÉDULA	NOMBRE	APELLIDO	GRADO	SECCION	NOTA	REPRESENTANTE	INGRESO		
15	Miguel	Suarez	1	A		Patricia Lopez	2023-03-17 17:25:39	EDITAR	DESINCORPORAR
479562	Yhon	Jeun	1	A		Daniela Castro	2023-03-17 17:25:39	EDITAR	DESINCORPORAR
6325784	Luisa	Palacios	1	A		Luzmary Bastidas	2023-03-17 17:25:39	EDITAR	DESINCORPORAR

❖ Carga de notas

Para cargar las notas, se debe seleccionar el grado y la sección, para ello se reutilizan de la clase `ControladorFormularios` y el método `ctrSeleccionarAula` quien solicita respuesta a la clase `modeloformulario` del modelo y el método `mdlSeleccionarAula`. Así se muestra la información del grado y sección seleccionada. Cada estudiante tiene la opción de cargar la nota, para ello se instancia el controlador `ctrCargarNota` el cual valida que se haya obtenido la variable `POST CargarNota` del formulario donde se solicita cargar el literal obtenido A, B, C, D o E, declara la variable `$tabla` y `$nota` tipo array con los valores del id y la nota, solicita respuesta al modelo el cual con una secuencia SQL `UPDATE` coloca el valor de nota en la tabla `estu` según el id y retorna OK.

❖ Generar reportes

En esta opción del menú es posible generar constancia de prosecución, informe final, certificado de educación primaria (FPDF).



CONCLUSIÓN

El sistema automatizado de notas (SAN) es un ejemplo básico de cómo desarrollar una aplicación web utilizando el patrón de diseño Modelo-Vista-Controlador (MVC). El sistema permite registrar usuarios y editar sus datos, para ello mediante la interacción entre la vista, el controlador y el modelo.

Para futuras mejoras del sistema, se recomienda implementar medidas de seguridad adicionales, como la validación de campos con expresiones regulares, la encriptación de contraseñas y la protección contra ataques de inyección SQL.

También se podría considerar la implementación de nuevas funcionalidades y la activación de perfiles de usuario (permisología) Todo ello con el fin de mejorar la funcionalidad y usabilidad del sistema automatizado de notas (SAN).