



República Bolivariana de Venezuela

Ministerio del Poder Popular para la Educación Universitaria, Ciencia y Tecnología

Universidad Politécnica Territorial del estado Mérida “Kléber Ramírez”

Programa Nacional de Formación en Informática

Mérida, estado Mérida

SISTEMA DE NÓMINA SISNO-18

MANUAL TECNICO DE SISTEMA

AUTOR:

Elber Alonso Rondón Hernández

PNFI T3-T1 Sección “B”

C.I: V.- 27919566

Mérida, enero de 2023

Sumario del manual

Presentación.....	5
Resumen.	5
Finalidad Del Manual.....	6
Introducción.	6
Aspectos teóricos.....	6
Diagramas De modelamiento	6
Aspecto técnico del desarrollo del sistema.....	6
Requerimientos del software.....	6
Aspectos técnicos.	7
Herramientas utilizadas para el desarrollo.....	7
SublimeText.....	7
GitHub.....	7
Node.js.....	8
TailwindCSS.....	8
Xampp.....	9
Diagramas de modelado.	10
Modelo entidad relacion.	10
Diccionario de datos.....	11
administrativos.....	11
carga_horaria.....	11
contactos.....	12
coordinacion	12
direcciones	12
directivo	13
docentes.....	13
empleados.....	14
estudios	14
informes	15



Manual técnico del sistema.

obreros.....	15
personal_doc.....	15
personas	16
telefonos	16
usuarios	17
Requerimientos del software.	18
Requisitos mínimos.....	18
Especificaciones dentro del código.	19
Estructura de directorios	19
Clases.	21
administrativo.php.....	22
Descripción de los métodos.....	23
carga-horaria.php.....	27
Descripción de los métodos	28
contacto.php.....	30
Descripción de los métodos	31
direccion.php.....	34
Descripción detallada de los métodos	35
docente.php.....	38
Descripción de los métodos	39
empleado.php.....	43
Descripción de los métodos	44
estudio.php	49
Descripción de los métodos	50
informe.php	53
Descripción de los métodos	54
obrero.php.....	57
Descripción de los métodos.....	58
persona.php.....	62
Descripción de los métodos	63
telefono.php	66
Descripción de los métodos	67



usuario.php	70
Descripción de los métodos	71
Controladores	75
base-datos.php	75
conexion.php	77
control-empleados.php	79
docente-excel.php.....	89
docente-pdf.php.....	92
login.php.....	95
logout.php	97
Vistas o pantalla.....	98
Diagrama general del sistema	98
Login.....	99
Menú principal.....	103
Consultar.....	107
Consulta de personal	110
Consulta de obreros.....	113
Consulta de docentes	116
Consulta de personal administrativo	119
Consulta de usuarios.....	122
Consultar empleado.....	125
Registrar	130
Paso 1	130
Paso 2	132
Editar.....	135



SISTEMA DE NÓMINA SISNO-18

MANUAL TECNICO DE SISTEMA

Presentación.

El siguiente manual se ha desarrollado con la finalidad de dar a conocer la información necesaria para realizar mantenimiento, instalación y exploración del software SISNO-18, el cual consta de diferentes módulos gestionar la nómina de un centro educativo.

El manual ofrece la información necesaria de ¿cómo está realizado el software? para que la persona (Desarrollador en PHP) que quiera editar el software lo haga de una manera apropiada, dando a conocer la estructura del desarrollo del sistema.

Resumen.

El presente manual detalla los aspectos técnicos e informáticos del software SISNO-18, con la finalidad de explicar la estructura del sistema al personal que quiera administrarlo, editarlo o configurarlo. La siguiente guía se encuentra dividida en las herramientas que se usaron para la creación del software con una breve explicación paso a paso, El sistema web maneja diferentes funcionalidades las cuales requieren de hardware y software el cual se explicará que funcionamiento realiza cada uno de ellos, dando sugerencias para el debido uso del sistema de información.



Manual técnico del sistema.

Finalidad Del Manual.

La finalidad de éste manual técnico es instruir a la persona que quiera administrar, editar o configurar el software SISNO-18 usando las debidas herramientas.

Introducción.

El manual se realiza con el fin de detallar el software SISNO-18 en términos técnicos para que la persona que vaya a administrar, editar o configurar el sistema lo haga de una manera apropiada. El documento se encuentra dividido en las siguientes secciones:

Aspectos teóricos: Se darán a conocer conceptos, definiciones y explicaciones de los componentes del sistema desde un punto de vista teórico para mayor entendimiento por parte del lector sobre el funcionamiento del sistema de información e herramientas.

Diagramas De modelamiento: Se compone por diagramas e ilustraciones alusivos al funcionamiento del sistema.

Aspecto técnico del desarrollo del sistema: Corresponde a la instrucción al lector sobre los componentes del sistema desde una perspectiva técnica en los aspectos de almacenamiento de datos, estructura del desarrollo y recomendaciones del uso debido del sistema.

Requerimientos del software: Detalla los requerimientos básicos necesarios para el funcionamiento del software.



Manual técnico del sistema.

Aspectos técnicos.

El sistema SISNO-18 tiene la finalidad de mejorar los procesos de gestión de personal dentro de la institución educativa. Se recomienda que el siguiente manual sea manipulado únicamente por la persona que quiera administrar, editar o configurar el software SISNO-18 para velar por la seguridad de los datos que se almacenan en la base de datos ya que pueden ser usados para otros fines.

Herramientas utilizadas para el desarrollo.

Es ésta sección se procede a explicar las herramientas informáticas empleadas para el desarrollo del sistema, los cuales se sugiere instalar para llevar a cabo modificaciones:

SublimeText.

SublimeText es un editor de texto el cual es caracterizado por sus amplias funcionalidades a la hora de desarrollar software, tiene incorporado la estructura en la sintaxis de diferentes lenguajes de programación con una interfaz amigable, en el desarrollo de SISNO-18 se trabajó con esta herramienta de marcado ya que indica en donde está ubicado los errores de sintaxis y la codificación se maneja de forma dinámica teniendo en cuenta que se requieren de paquetes adicionales.

GitHub.

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

GitHub provee de funcionalidades para hacer un fork y solicitar pulls; así como también puede llevar seguimiento de los cambios.



Manual técnico del sistema.

Con SISNO-18, se maneja un repositorio público propio alojado en:

- https://github.com/rondon18/sistema_ing_soft_II

Cabe mencionar que en este mismo se encuentra una breve descripción del proyecto, así como una guía breve para su instalación.

Node.js.

Node.js es un entorno de tiempo de ejecución de JavaScript (de ahí su terminación en .js haciendo alusión al lenguaje JavaScript). Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript.

No obstante, su principal uso en el desarrollo del sistema ha sido para uso de TailwindCSS y sus dependencias.

TailwindCSS.

Tailwind CSS es un framework de CSS de código abierto para el diseño de páginas web. La principal característica de esta biblioteca es que, a diferencia de otras como Bootstrap, no genera una serie de clases predefinidas para elementos como botones o tablas.

Fue usado para el maquetado del proyecto. Gracias a tailwind CLI, el cual permite, en conjunto con NPM de Node.js, identificar que clases del framework se están usando para generar un compilado que incluya solo las clases usadas para mantener cortos los tiempos de carga en conexiones lentas.



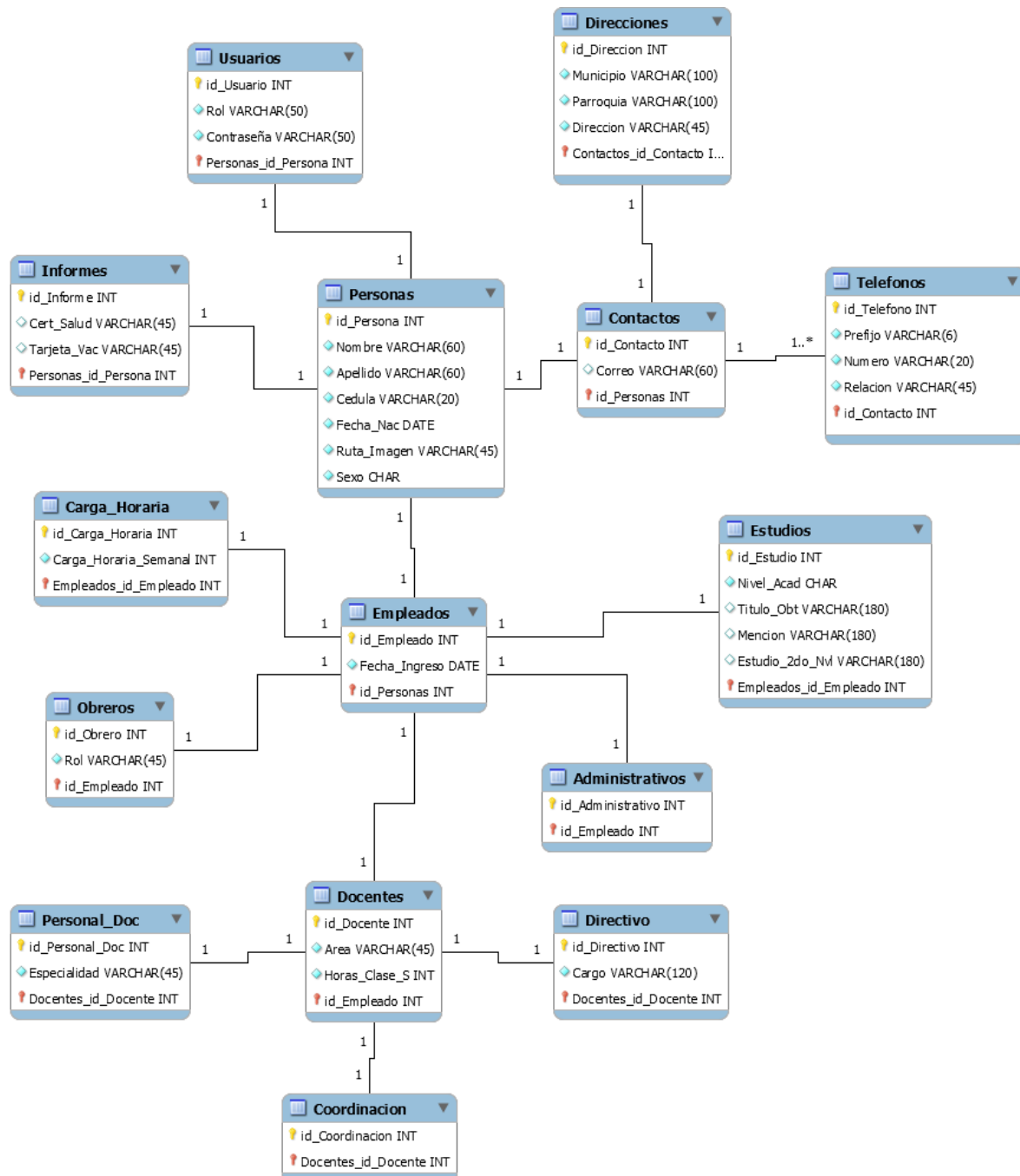
Manual técnico del sistema.

Xampp.

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. El nombre es en realidad un acrónimo: X, Apache, MariaDB/MySQL, PHP, Perl.

Diagramas de modelado.

Modelo entidad relacion.





Manual técnico del sistema.

Diccionario de datos.

Para el almacenamiento de datos del software, se definen los campos necesarios para cada una de las entidades relacionadas con el aplicativo.

administrativos.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Administrativo (<i>Primaria</i>)	int(11)	No		
id_Empleado (<i>Primaria</i>)	int(11)	No		empleados -> id_Empleado

Índices.

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Administrativo	No
				id_Empleado	No
fk_Administrativo_Empleado1_idx	BTREE	No	No	id_Empleado	No

carga_horaria.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Carga_Horaria (<i>Primaria</i>)	int(11)	No		
Carga_Horaria_Semanal	int(11)	No		
Empleados_id_Empleado (<i>Primaria</i>)	int(11)	No		empleados -> id_Empleado

Índices.

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Carga_Horaria	No
				Empleados_id_Empleado	No
fk_Carga_Horaria_Empleados1_idx	BTREE	No	No	Empleados_id_Empleado	No



Manual técnico del sistema.

contactos.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Contacto (<i>Primaria</i>)	int(11)	No		
Correo	varchar(60)	Sí	<i>NULL</i>	
id_Personas (<i>Primaria</i>)	int(11)	No		personas -> id_Persona

Índices.

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Contacto	No
				id_Personas	No
fk_Contactos_Personas1_idx	BTREE	No	No	id_Personas	No

coordinacion.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Coordinacion (<i>Primaria</i>)	int(11)	No		
Docentes_id_Docente (<i>Primaria</i>)	int(11)	No		docentes -> id_Docente

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Coordinacion	No
				Docentes_id_Docente	No
fk_Coordinacion_Docentes1_idx	BTREE	No	No	Docentes_id_Docente	No

direcciones.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Direccion (<i>Primaria</i>)	int(11)	No		
Municipio	varchar(100)	No		
Parroquia	varchar(100)	No		
Direccion	varchar(45)	No		
Contactos_id_Contacto (<i>Primaria</i>)	int(11)	No		contactos -> id_Contacto

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Direccion	No
				Contactos_id_Contacto	No
fk_Direccion>Contactos1_idx	BTREE	No	No	Contactos_id_Contacto	No

directivo.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Directivo <i>(Primaria)</i>	int(11)	No		
Cargo	varchar(120)	No		
Docentes_id_Docente <i>(Primaria)</i>	int(11)	No		docentes -> id_Docente

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Directivo	No
				Docentes_id_Docente	No
fk_Directivo_Docentes1_idx	BTREE	No	No	Docentes_id_Docente	No

docentes.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Docente <i>(Primaria)</i>	int(11)	No		
Area	varchar(45)	No		
Horas_Clase_S	int(11)	No		
id_Empleado <i>(Primaria)</i>	int(11)	No		empleados -> id_Empleado

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Docente	No
				id_Empleado	No
fk_Docente_Empleado1_idx	BTREE	No	No	id_Empleado	No



Manual técnico del sistema.

empleados.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Empleado (<i>Primaria</i>)	int(11)	No		
Fecha_Ingreso	date	No		
id_Personas (<i>Primaria</i>)	int(11)	No		personas -> id_Persona

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Empleado	No
				id_Personas	No
fk_Empleado_Personas1_idx	BTREE	No	No	id_Personas	No

estudios.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Estudio (<i>Primaria</i>)	int(11)	No		
Nivel_Acad	char(1)	No		
Titulo_Obt	varchar(180)	Sí	NULL	
Mencion	varchar(180)	Sí	NULL	
Estudio_2do_Nvl	varchar(180)	Sí	NULL	
Empleados_id_Empleado (<i>Primaria</i>)	int(11)	No		empleados -> id_Empleado

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Estudio	No
				Empleados_id_Empleado	No
fk_Estudio_Empleados1_idx	BTREE	No	No	Empleados_id_Empleado	No



Manual técnico del sistema.

informes.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Informe (<i>Primaria</i>)	int(11)	No		
Cert_Salud	varchar(45)	Sí	NULL	
Tarjeta_Vac	varchar(45)	Sí	NULL	
Personas_id_Persona (<i>Primaria</i>)	int(11)	No		personas -> id_Persona

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Informe	No
				Personas_id_Persona	No
fk_Informe_Personas1_idx	BTREE	No	No	Personas_id_Persona	No

obreros.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Obrero (<i>Primaria</i>)	int(11)	No		
Rol	varchar(45)	No		
id_Empleado (<i>Primaria</i>)	int(11)	No		empleados -> id_Empleado

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Obrero	No
				id_Empleado	No
fk_Obrero_Empleado1_idx	BTREE	No	No	id_Empleado	No

personal_doc.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Personal_Doc (<i>Primaria</i>)	int(11)	No		
Especialidad	varchar(45)	No		
Docentes_id_Docente (<i>Primaria</i>)	int(11)	No		docentes -> id_Docente

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Personal_Doc	No
				Docentes_id_Docente	No
fk_Personal_Doc_Docentes1_idx	BTREE	No	No	Docentes_id_Docente	No

personas.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Persona <i>(Primaria)</i>	int(11)	No		
Nombre	varchar(60)	No		
Apellido	varchar(60)	No		
Cedula	varchar(20)	No		
Fecha_Nac	date	No		
Ruta_Imagen	varchar(45)	No		
Sexo	char(1)	No		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Persona	No
Cedula_UNIQUE	BTREE	Sí	No	Cedula	No

telefonos.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Telefono <i>(Primaria)</i>	int(11)	No		
Prefijo	varchar(6)	No		
Numero	varchar(20)	No		
Relacion	varchar(45)	No		
id_Contacto <i>(Primaria)</i>	int(11)	No		contactos -> id_Contacto

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Telefono	No
				id_Contacto	No
fk_Telefonos_Contactos1_idx	BTREE	No	No	id_Contacto	No

usuarios.

Columna	Tipo	Nulo	Predeterminado	Enlaces a
id_Usuario (<i>Primaria</i>)	int(11)	No		
Rol	varchar(50)	No		
Contraseña	varchar(50)	No		
Personas_id_Persona (<i>Primaria</i>)	int(11)	No		personas -> id_Persona

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Nulo
PRIMARY	BTREE	Sí	No	id_Usuario	No
				Personas_id_Persona	No
fk_Rol_Personas1_idx	BTREE	No	No	Personas_id_Persona	No



Manual técnico del sistema.

Requerimientos del software.

En esta sección se detallará los requisitos mínimos del sistema para poder ejecutar los aplicativos usados para modificar el software SISNO-18.

Requisitos mínimos.

- Sistema Operativo: Windows 7 de 64 bits o distribución GNU/Linux de 64 bits
- Procesador: Intel Core Celeron
- Memoria RAM: 1GB
- Disco Duro: 1GB



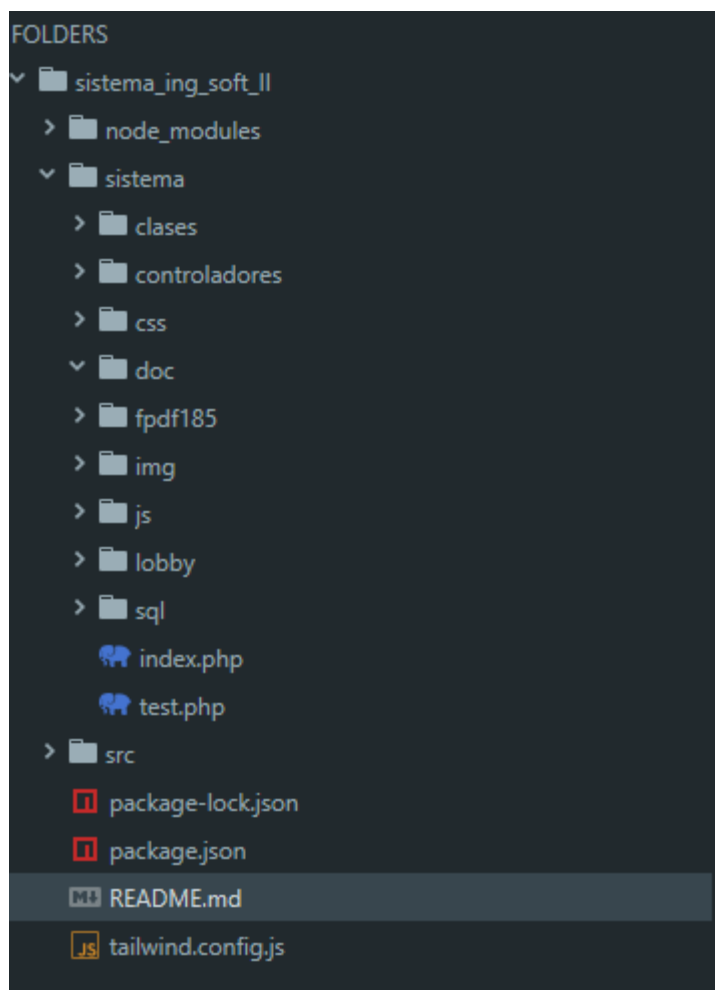
Manual técnico del sistema.

Especificaciones dentro del código.

Estructura de directorios.

Los directorios dentro del sistema se encuentran organizados de la siguiente manera.

Teniendo como directorio raíz sistema_ing_soft_II.



Los subdirectorios principales del sistema vienen siendo descritos de la siguiente manera:

- **node_modules:** abarca los módulos de node.js que se están usando y son necesarios a la hora de hacer alguna modificación de interfaz.



Manual técnico del sistema.

- **sistema:** contiene los archivos que componen al sistema.
 - **clases:** contiene archivos php con las clases que se usan en el sistema. Uno para cada clase (Véase diccionario de datos).
 - **controladores:** contiene archivos con los controladores que permiten el flujo de los datos entre interfaces y las clases del sistema.
 - **css:** incluye las hojas de estilos de cascada del sistema
 - **doc:** contiene el o los manuales del sistema.
 - **fpdf185:** contiene los archivos de fpdf necesarios para su funcionamiento.
 - **img:** contiene imágenes del sistema y subidas por los usuarios.
 - **js:** contiene archivos y librerías de javascript.
 - **lobby:** contiene las vistas del sistema. A excepción de inicio de sesión.
 - **sql:** contiene archivos sql para cargar la base de datos base, así como para cargar datos de prueba.
- **src:** contiene el archivo principal de tailwindcss, con sus componentes y clases para poder ser compiladas.



Manual técnico del sistema.

Clases.

El sistema cuenta con un total de 12 clases y una interfaz. Una por cada tabla existente en la base de datos, exceptuando la interfaz cálculos.

De igual manera, cada una de estas clases cuenta con los métodos getter y setter para cada uno de sus atributos

Cabe mencionar que dentro de las clases, estos métodos que no son ni getters ni setters se encuentran descritos en comentarios.

```
// INTERFAZ
calculos.php

// CLASES
administrativo.php
carga-horaria.php
contacto.php
direccion.php
docente.php
empleado.php
estudio.php
informe.php
obrero.php
persona.php
telefono.php
usuario.php
```

A continuación, se pasa a describir individualmente las clases y la interfaz del sistema. Los métodos que no son getters (retornar valor de atributo) o setters (asignar valor a atributo) serán descritos a mayor detalle.

Se debe resaltar que en el caso de las clases, se requiere que esté en uso o incluido el controlador conexion.php

administrativo.php

Clase padre:	Empleado, Persona
Nombre de la clase / interfaz:	Administrativo
Atributos:	
id_Administrativo	Privado. Id que referencia al Administrativo en base de datos
Métodos:	
__construct	Constructor de la clase.
insertarAdministrativo	Público. Registra al Administrativo en la base de datos
editarAdministrativo	Público. Edita los valores del Administrativo en la base de datos
mostrar	Público. Retorna una lista con todos los Administrativos registrados en la base de datos
consultar	Público. Retorna los valores de un administrativo en concreto. Recibe el id de una persona como parámetro.



Manual técnico del sistema.

Descripción de los métodos.

insertarAdministrativo:

```
// Inserta un empleado administrativo
public function insertarAdministrativo() {

    $this->insertarPersona();
    $this->insertarEmpleado();

    $conexion = conectar();

    $id_Empleado = $this->getid_Empleado();

    $sql = "
        INSERT IGNORE INTO `administrativos`
        (
            `id_Administrativo`,
            `id_Empleado`
        )
        VALUES
        (
            NULL,
            '$id_Empleado'
        );
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setid_Administrativo($conexion->insert_id);

    desconectar($conexion);
}
```

Ejecuta los métodos heredados insertarPersona e insertarEmpleado. Luego realiza una conexión con la base de datos, Se genera una variable local con el id del empleado (debe haberse establecido el atributo antes de ejecutarlo).

Guarda en una variable la instrucción SQL para insertar y luego la ejecuta o lanza un mensaje de error en caso de fallar.

Asigna el atributo de id_Administrativo con el valor del registro realizado y cierra la conexión a base de datos.

editarAdministrativo:

```
// Edita un empleado administrativo en especifico
public function editarAdministrativo() {

    $this->editarPersona();
    $this->editarEmpleado();

    // $conexion = conectar();

    // $conexion->query($sql) or die("error: ".$conexion->error);
    // desconectar($conexion);
}
```

Este método ejecuta los métodos heredados editarPersona y editarEmpleado.

Debido a que el personal administrativo no tiene un atributo exclusivo de su clase, solo se estarían editando sus atributos heredados.

mostrar:

```
// Retorna una lista con todos los empleados que sean administrativos
public function mostrar(){
    $conexion = conectar();

    $sql = "
    SELECT * FROM `personas`,`empleados`,`administrativos`
    WHERE
    `personas`.`id_Persona` = `empleados`.`id_Personas`
    AND
    `empleados`.`id_Empleado` = `administrativos`.`id_Empleado`
    ";

    $fetch = $conexion->query($sql);

    $resultados = $fetch->fetch_all(MYSQLI_ASSOC);

    $administrativos = [];
    foreach ($resultados as $administrativo) {
        $administrativos[] = $administrativo;
    }

    desconectar($conexion);

    return $administrativos;
}
```

Este método usa polimorfismo, y se diferencia del método presente en su clase padre.



Manual técnico del sistema.

Inicia una conexión a base de datos y posteriormente guarda la instrucción SQL en una variable, este mismo realiza una consulta que trae datos de personas, empleados y administrativos, en los cuales se cumplan estas condiciones: el id de la persona debe coincidir con el id de persona almacenado en la tabla empleados, asimismo el id de empleado debe coincidir con el almacenado en la tabla administrativos.

Luego, ejecuta la consulta para traer todos los registros que cumplen con las condiciones; estos mismos serán almacenados uno a uno en un arreglo. Cerrará la conexión y retornará la lista de empleados.

consultar.

```
public function consultar($id_Persona) {
    $conexion = conectar();

    $sql = "
    SELECT * FROM `personas`,`empleados`,`administrativos`
    WHERE
        `personas`.`id_Persona` = '$id_Persona'
    AND
        `empleados`.`id_Personas` = '$id_Persona'
    AND
        `empleados`.`id_Empleado` = `administrativos`.`id_Empleado`
    ";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_assoc();

    desconectar($conexion);

    return $resultado;
}
```

Inicialmente este método inicia una conexión a base de datos, almacena la consulta SQL en una variable, esta misma realiza una consulta que trae datos de una persona, empleado y administrativo, en el cual se cumplan estas condiciones: el id de la persona debe coincidir con el id especificado como parámetro, de igual modo el id de persona almacenado en la tabla



Manual técnico del sistema.

empleados, asimismo el id de empleado debe coincidir con el almacenado en la tabla administrativos.

Se ejecutará la consulta y traerá los datos del empleado como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos del empleado.

carga-horaria.php

Clase padre:	No hereda
Nombre de la clase / interfaz:	Carga_Horaria
Atributos:	
id_Carga_Horaria	Privado. Id asociado a al registro de carga horaria en la base de datos
Carga_Horaria_Semanal	Privado. Número de horas semanales que se cumplen
Empleados_id_Empleado	Privado. Id del empleado al que está asociada la carga horaria
Métodos:	
__construct	Constructor de la clase.
insertarCarga_Horaria	Público. Inserta la carga horaria en la base de datos.
editarCarga_Horaria	Público Edita los valores de carga horaria en la base de datos
consultar	Público. Retorna los valores de carga horaria de un empleado en concreto. Recibe el id de una persona como parámetro.



Manual técnico del sistema.

Descripción de los métodos:

insertarCarga_Horaria:

```
// Inserta la carga horaria semanal de un empleado
public function insertarCarga_Horaria() {
    $conexion = conectar();

    $Carga_Horaria_Semanal = $this->getCarga_Horaria_Semanal();
    $Empleados_id_Empleado = $this->getEmpleados_id_Empleado();

    $sql = "INSERT IGNORE INTO `carga_horaria`(`id_Carga_Horaria`,
        `Carga_Horaria_Semanal`, `Empleados_id_Empleado`) VALUES (
            NULL,
            '$Carga_Horaria_Semanal',
            '$Empleados_id_Empleado'
        )";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setid_Carga_Horaria($conexion->insert_id);

    desconectar($conexion);
}
```

Comienza iniciando una conexión a base de datos, luego guarda en variables locales la carga horaria semanal y el id de empleado.

Prepara la sentencia SQL de inserción y la ejecuta, o muestra error en caso de fallar.

Finalmente almacena como atributo el id del registro realizado y cierra la conexión a base de datos.



Manual técnico del sistema.

editarCarga_Horaria:

```
// Edita la carga horaria semanal de un empleado
public function editarCarga_Horaria() {
    $conexion = conectar();

    $Carga_Horaria_Semanal = $this->getCarga_Horaria_Semanal();
    $Empleados_id_Empleado = $this->getEmpleados_id_Empleado();

    $sql = "UPDATE `carga_horaria` SET
    `Carga_Horaria_Semanal`='{$Carga_Horaria_Semanal}'
    WHERE
    `Empleados_id_Empleado`='{$Empleados_id_Empleado}'
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    desconectar($conexion);
}
```

Comienza iniciando una conexión a base de datos, luego guarda en variables locales la carga horaria semanal y el id de empleado.

Prepara la sentencia SQL de edición y la ejecuta, o muestra error en caso de fallar.

Finalmente cierra la conexión a base de datos.

consultar:

```
// Consulta la carga horaria semanal de un empleado en especifico
public function consultar($id_Empleado) {
    $conexion = conectar();

    $sql = "SELECT * FROM `carga_horaria` WHERE `Empleados_id_Empleado`='{$id_Empleado}'";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_assoc();

    desconectar($conexion);

    return $resultado;
}
```

Comienza iniciando una conexión a base de datos, luego prepara la sentencia SQL de consulta usando el id del empleado como criterio, y la ejecuta, o muestra error en caso de fallar.

Finalmente cierra la conexión a base de datos.

contacto.php

Clase padre:	No hereda
Nombre de la clase / interfaz:	Contacto
Atributos:	
id_Contacto	Privado. Id de la ficha de contacto en la base de datos.
Correo	Privado. Correo electrónico de la persona.
id_Personas	Privado. Id de la persona a la que está asociada la ficha de contacto.
T_principal	Privado. Teléfono principal de la persona.
T_secundario	Privado. Teléfono principal de la persona.
T_auxiliar	Privado. Teléfono principal de la persona.
Métodos:	
__construct	Constructor de la clase.
insertarContacto	Público. Inserta la ficha de contacto en la base de datos.
editarContacto	Público. Edita los valores de la ficha de contacto en la base de datos
consultar	Público. Retorna la ficha de contacto de una persona en concreto. Recibe como parámetro el id de persona



Manual técnico del sistema.

Descripción de los métodos.

insertarContacto:

```
// Inserta la ficha de contacto de una persona
public function insertarContacto() {
    $conexion = conectar();

    $Correo = $this->getCorreo();
    $id_Personas = $this->getid_Personas();

    $sql = "
    INSERT IGNORE INTO `contactos`
    (
        `id_Contacto`,
        `Correo`,
        `id_Personas`)
    VALUES
    (
        NULL,
        '$Correo',
        '$id_Personas'
    );
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setid_Contacto($conexion->insert_id);

    desconectar($conexion);
}
```

El método inicia una conexión a base de datos, almacena el correo e id de persona en variables locales y prepara la sentencia SQL a ejecutar. Luego se ejecuta la orden, en caso de fallar mostrará un error.

Al finalizar, almacena como atributo el id de la ficha de contacto registrada y cierra la conexión a base de datos.

editarContacto:

```
// Edita la ficha de contacto de una persona
public function editarContacto() {
    $conexion = conectar();

    $Correo = $this->getCorreo();
    $id_Personas = $this->getid_Personas();

    $sql = "
        UPDATE
            `contactos`
        SET
            `Correo` = '$Correo'
        WHERE
            `id_Personas` = '$id_Personas'
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    desconectar($conexion);
}
```

El método inicia una conexión a base de datos, almacena el correo e id de persona en variables locales y prepara la sentencia SQL con los nuevos valores. Luego se ejecuta la orden, en caso de fallar mostrará un error.

Al finalizar, cierra la conexión a base de datos.

consultar:

```
// Consulta la ficha de contacto de una persona en concreto
public function consultar($id_Personas) {
    $conexion = conectar();

    $sql = "SELECT * FROM `contactos` WHERE `id_Personas` = '$id_Personas'";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_assoc();

    desconectar($conexion);

    return $resultado;
}
```




Manual técnico del sistema.

El método inicia una conexión a base de datos, prepara la sentencia SQL usando el valor de su parámetro como criterio de búsqueda. Luego se ejecuta la orden, en caso de fallar mostrará un error.

Entonces, traerá lo datos de la ficha de contacto como un arreglo asociativo. Al finalizar, cierra la conexión a base de datos y retorna los datos obtenidos.

direccion.php

Clase padre:	No hereda
Nombre de la clase / interfaz:	Direccion
Atributos:	
id_Direccion	Privado. Id del registro de dirección en la base de datos.
Municipio	Privado. Municipio en que reside el empleado.
Parroquia	Privado. Parroquia en que reside el empleado.
Direccion	Privado. Dirección en que reside el empleado.
Contactos_id_Contacto	Privado. Id de la ficha de contacto a la que está asociada la dirección
Métodos:	
__construct	Constructor de la clase.
insertarDireccion	Público Registra la dirección de un empleado.
editarDireccion	Público Actualiza la dirección de un usuario.
consultar	Público Consulta la dirección de un usuario. Recibe como parámetro el id de contacto.



Manual técnico del sistema.

Descripción detallada de los métodos.

insertarDireccion:

```
// inserta la direccion de una persona
public function insertarDireccion() {
    $conexion = conectar();

    $Municipio = $this->getMunicipio();
    $Parroquia = $this->getParroquia();
    $Direccion = $this->getDireccion();

    $id_Contacto = $this->getContactos_id_Contacto();

    $sql = "
    INSERT IGNORE INTO `direcciones`(`id_Direccion`, `Municipio`, `Parroquia`, `
    Direccion`, `Contactos_id_Contacto`)
    VALUES (
        NULL,
        '$Municipio',
        '$Parroquia',
        '$Direccion',
        '$id_Contacto'
    );
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setid_Direccion($conexion->insert_id);

    desconectar($conexion);
}
```

El método almacena en variables locales el municipio, parroquia, dirección e id de la ficha de contacto del empleado, prepara la sentencia SQL para insertar esos datos y la ejecuta, de haber un error se mostrará, finalmente almacena como atributo el id de la dirección insertada y cierra la conexión con la base de datos.

Manual técnico del sistema.

editarDireccion:

```
// edita la direccion de una persona
public function editarDireccion() {
    $conexion = conectar();

    $Municipio = $this->getMunicipio();
    $Parroquia = $this->getParroquia();
    $Direccion = $this->getDireccion();

    $id_Contacto = $this->getContactos_id_Contacto();

    $sql = "UPDATE `direcciones` SET
    `Municipio`='$Municipio',
    `Parroquia`='$Parroquia',
    `Direccion`='$Direccion'
    WHERE
    `Contactos_id_Contacto`='$id_Contacto';";

    $conexion->query($sql) or die("error: ".$conexion->error);
    desconectar($conexion);
}
```

El método almacena en variables locales el municipio, parroquia, dirección e id de la ficha de contacto del empleado, prepara la sentencia SQL para editar esos datos y la ejecuta, de haber un error se mostrará, finalmente cierra la conexión con la base de datos.

consultar:

```
// consulta la direccion de una persona
public function consultar($id_Contacto) {
    $conexion = conectar();

    $sql = "SELECT * FROM `direcciones` WHERE `Contactos_id_Contacto`='
    $id_Contacto';";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_assoc();

    desconectar($conexion);

    return $resultado;
}
```



Manual técnico del sistema.

El método inicia una conexión a base de datos, prepara la sentencia SQL usando el valor de su parámetro como criterio de búsqueda. Luego se ejecuta la orden, en caso de fallar mostrará un error.

Entonces, traerá lo datos de la dirección como un arreglo asociativo. Al finalizar, cierra la conexión a base de datos y retorna los datos obtenidos.

docente.php

Clase padre:	Empleado, Persona
Nombre de la clase / interfaz:	Docente
Atributos:	
id_Docente	Privado. Id del docente en la base de datos
Horas_Clase_S	Privado. Horas de clase que imparte el docente por semana
Area	Privado. Área en que imparte clases.
Métodos:	
__construct	Constructor de la clase.
insertarDocente	Público. Registra un docente en la base de datos.
editarDocente	Público. Edita a un docente en la base de datos
mostrar	Público. Muestra los docentes registrados en la base de datos.
consultar	Público. Consulta los datos de un docente. Recibe el id de persona como parámetro.



Manual técnico del sistema.

Descripción de los métodos:

insertarDocente:

```
public function insertarDocente() {  
  
    $this->insertarPersona();  
    $this->insertarEmpleado();  
  
    $conexion = conectar();  
  
    $Horas_Clase_S = $this->getHoras_Clase_S();  
    $Area = $this->getArea();  
    $id_Empleado = $this->getid_Empleado();  
  
    $sql = "  
        INSERT IGNORE INTO `docentes` (  
            `id_Docente`,  
            `Area`,  
            `Horas_Clase_S`,  
            `id_Empleado`  
        )  
        VALUES(  
            NULL,  
            '$Horas_Clase_S',  
            '$Area',  
            '$id_Empleado'  
        );  
    ";  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    $this->setid_Docente($conexion->insert_id);  
  
    desconectar($conexion);  
}
```

El método inicia utilizando los métodos heredados insertarPersona e insertarEmpleado, luego inicia una conexión a base de datos, almacena en variables locales las horas de clase impartidas por el docente, su área e id de empleado, prepara la sentencia SQL para insertar, luego la ejecuta o muestra un mensaje de error en caso de fallar, finalmente toma el id del registro creado y cierra la conexión.

editarDocente:

```
public function editarDocente() {  
  
    $this->editarPersona();  
    $this->editarEmpleado();  
  
    $conexion = conectar();  
  
    $Horas_Clase_S = $this->getHoras_Clase_S();  
    $Area = $this->getArea();  
    $id_Empleado = $this->getid_Empleado();  
  
    $sql = "  
        UPDATE  
        `docentes`  
        SET  
        `Horas_Clase_S`='$Horas_Clase_S',  
        `Area`='$Area'  
        WHERE  
        `id_Empleado`='$id_Empleado'  
    ";  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    desconectar($conexion);  
}
```

El método inicia usando los métodos heredados editarPersona y editarEmpleado, luego inicia una conexión a base de datos, almacena en variables locales las horas de clase impartidas por el docente, su área e id de empleado, prepara la sentencia SQL con los nuevos valores para editar, luego la ejecuta o muestra un mensaje de error en caso de fallar, finalmente cierra la conexión.



Manual técnico del sistema.

mostrar:

```
public function mostrar(){
    $conexion = conectar();

    $sql = "
    SELECT * FROM `personas`,`empleados`,`docentes`
    WHERE
        `personas`.`id_Persona` = `empleados`.`id_Personas`
    AND
        `empleados`.`id_Empleado` = `docentes`.`id_Empleado`
    ";

    $fetch = $conexion->query($sql);

    $resultados = $fetch->fetch_all(MYSQLI_ASSOC);

    $docentes = [];
    foreach ($resultados as $docente) {
        $docentes[] = $docente;
    }

    desconectar($conexion);

    return $docentes;
}
```

Este método usa polimorfismo, y se diferencia del método presente en su clase padre.

Inicia una conexión a base de datos y posteriormente guarda la instrucción SQL en una variable, este misma realiza una consulta que trae datos de personas, empleados y docentes, en los cuales se cumplan estas condiciones: el id de la persona debe coincidir con el id de persona almacenado en la tabla empleados, asimismo el id de empleado debe coincidir con el almacenado en la tabla docentes.

Luego, ejecuta la consulta para traer todos los registros que cumplen con las condiciones; estos mismos serán almacenados uno a uno en un arreglo. Cerrará la conexión y retornará la lista de empleados.

Manual técnico del sistema.

consultar:

```
public function consultar($id_Persona) {  
    $conexion = conectar();  
  
    $sql = "  
    SELECT * FROM `personas`,`empleados`,`docentes`  
    WHERE  
        `personas`.`id_Persona` = '$id_Persona'  
    AND  
        `empleados`.`id_Personas` = '$id_Persona'  
    AND  
        `empleados`.`id_Empleado` = `docentes`.`id_Empleado`  
    ;";  
  
    $fetch = $conexion->query($sql);  
  
    $resultado = $fetch->fetch_assoc();  
  
    desconectar($conexion);  
  
    return $resultado;  
}
```

Inicialmente este método inicia una conexión a base de datos, almacena la consulta SQL en una variable, esta misma realiza una consulta que trae datos de una persona, empleado y docente, en el cual se cumplan estas condiciones: el id de la persona debe coincidir con el id especificado como parámetro, de igual modo el id de persona almacenado en la tabla empleados, asimismo el id de empleado debe coincidir con el almacenado en la tabla docente.

Se ejecutará la consulta y traerá los datos del empleado como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos del empleado.

empleado.php

Clase padre:	Persona
Nombre de la clase / interfaz:	Empleado
Atributos:	
id_Empleado	Privado. Id del empleado en la base de datos
Fecha_Ingreso	Privado. Fecha de ingreso a nómina del empleado.
Métodos:	
__construct	Constructor de la clase.
insertarEmpleado	Público. Registra un empleado en la base de datos.
editarEmpleado	Público. Edita un empleado en la base de datos
mostrar	Público. Retorna una lista con todos los empleados registrados.
verificarTipo	Público. Verifica si el empleado es obrero, docente o administrativo.
contarEmpleados	Público. Retorna el número de empleados registrados.

Manual técnico del sistema.

Descripción de los métodos.

insertarEmpleado:

```
public function insertarEmpleado() {  
    $conexion = conectar();  
  
    $Fecha_Ingreso = $this->getFecha_Ingreso();  
    $id_Personas = $this->getid_Personas();  
  
    $sql = "INSERT IGNORE INTO `empleados`(`id_Empleado`, `Fecha_Ingreso`, `  
        id_Personas`) VALUES (  
            NULL,  
            '$Fecha_Ingreso',  
            '$id_Personas'  
        );";  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    $this->setid_Empleado($conexion->insert_id);  
    desconectar($conexion);  
}
```

El método inicia una conexión con la base de datos, luego almacena la fecha de ingreso a nómina y el id de persona en variables locales, luego prepara la sentencia SQL para insertar, la ejecuta y muestra un mensaje de error en caso de fallar. Finalmente, guarda como atributo el id del empleado registrado y cierra la conexión a base de datos

editarEmpleado:

```
public function editarEmpleado() {  
    $conexion = conectar();  
  
    $Fecha_Ingreso = $this->getFecha_Ingreso();  
    $id_Personas = $this->getid_Personas();  
  
    // La cédula no es actualizable  
    $sql = "  
    UPDATE `empleados` SET  
    `Fecha_Ingreso`='$Fecha_Ingreso'  
    WHERE  
    `id_Personas`='$id_Personas'  
    ";  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    desconectar($conexion);  
}
```

El método inicia una conexión con la base de datos, luego almacena la fecha de ingreso a nómina y el id de persona en variables locales, luego prepara la sentencia SQL para editar los valores actuales, la ejecuta y muestra un mensaje de error en caso de fallar. Finalmente, cierra la conexión a base de datos



Manual técnico del sistema.

mostrar:

```
public function mostrar() {  
    $conexion = conectar();  
  
    $sql = "SELECT * FROM `personas`, `empleados` WHERE `personas`.`id_Persona` = `  
        empleados`.`id_Personas`";  
  
    $fetch = $conexion->query($sql);  
  
    $resultados = $fetch->fetch_all(MYSQLI_ASSOC);  
  
    $empleados = [];  
    foreach ($resultados as $empleado) {  
        $empleados[] = $empleado;  
    }  
  
    desconectar($conexion);  
  
    return $empleados;  
}
```

El método inicia una conexión a base de datos, luego prepara la sentencia SQL, la cual consulta todos las personas que sean empleados y trae los datos de ambas tablas, luego anexa todos estos resultados a un arreglo de arreglos, cierra la conexión y retorna los resultados obtenidos.

verificarTipo:

```
public function verificarTipo() {
    $conexion = conectar();

    $id_Empleado = $this->getid_Empleado();

    // Consulta obrero
    $sql = "SELECT COUNT(*) as 'Cuenta' FROM `empleados`, `obreros` WHERE `empleados`.`id_Empleado` = '$id_Empleado' AND `obreros`.`id_Empleado` = '$id_Empleado'";
    $ob = $conexion->query($sql);
    $ob = $ob->fetch_assoc();

    // Consulta docente
    $sql = "SELECT COUNT(*) as 'Cuenta' FROM `empleados`, `docentes` WHERE `empleados`.`id_Empleado` = '$id_Empleado' AND `docentes`.`id_Empleado` = '$id_Empleado'";
    $doc = $conexion->query($sql);
    $doc = $doc->fetch_assoc();

    // Consulta administrativo
    $sql = "SELECT COUNT(*) as 'Cuenta' FROM `empleados`, `administrativos` WHERE `empleados`.`id_Empleado` = '$id_Empleado' AND `administrativos`.`id_Empleado` = '$id_Empleado'";
    $adm = $conexion->query($sql);
    $adm = $adm->fetch_assoc();

    desconectar($conexion);

    // Dependiendo de cual de las 3 retorna un valor

    // Obrero
    if ($ob['Cuenta'] == '1' and ($doc['Cuenta'] == '0' and $adm['Cuenta'] == '0')) {
        $val = 0;
    }
    // Docente
    elseif ($doc['Cuenta'] == '1' and ($ob['Cuenta'] == '0' and $adm['Cuenta'] == '0')) {
        $val = 1;
    }
    // Administrativo
    elseif ($adm['Cuenta'] == '1' and ($doc['Cuenta'] == '0' and $ob['Cuenta'] == '0')) {
        $val = 2;
    }
    // Otro
    else {
        $val = 3;
    }

    return $val;
}
```

El método inicia una conexión con la base de datos, luego hace tres consultas individuales, en cada una verifica si el empleado es un obrero, un docente o un administrativo mediante un conteo.

Luego, valida a qué tipo de empleado es en base a los resultados y retornara un valor: 0 para obrero, 1 para docente, 2 para administrativo, en caso de tener múltiples cargos o no poseer ninguno, retorna 3.



Manual técnico del sistema.

contarEmpleados:

```
public function contarEmpleados() {  
    $conexion = conectar();  
  
    $sql = "SELECT COUNT(*) FROM `personas`, `empleados` WHERE `personas`.`id_Persona` = `empleados`  
        `.id_Personas`";  
  
    $fetch = $conexion->query($sql);  
  
    $resultados = $fetch->fetch_all(MYSQLI_ASSOC);  
  
    $conteo = $resultados[0]["COUNT(*)"];  
  
    desconectar($conexion);  
  
    return $conteo;  
}
```

El método inicia una conexión con la base de datos, prepara una sentencia SQL en la que cuenta cuantos empleados hay registrados, conteo que es almacenado en una variable local, se cierra la conexión a base de datos y se retorna el conteo de empleados registrados.

estudio.php

Clase padre:	No hereda
Nombre de la clase / interfaz:	Estudio
Atributos:	
id_Estudio	Privado. Id de los datos de estudio en la base de datos
Nivel_Acad	Privado. Nivel académico del empleado.
Titulo_Obt	Privado. Título académico del empleado.
Mencion	Privado. Mención del título del empleado.
Estudio_2do_Nvl	Privado. Estudio de segundo nivel que posee el empleado.
Empleados_id_Empleado	Privado. Id del empleado al que está asociado.
Métodos:	
__construct	Constructor de la clase.
insertarEstudio	Público. Inserta los datos de estudios a la base de datos.
editarEstudio	Público. Edita los datos de estudios en la base de datos
consultar	Público. Retorna los datos de estudios de un empleado

Manual técnico del sistema.

Descripción de los métodos.

insertarEstudio:

```
public function insertarEstudio() {
    $conexion = conectar();

    $Nivel_Acad = $this->getNivel_Acad();
    $Titulo_Obt = $this->getTitulo_Obt();
    $Mencion = $this->getMencion();
    $Estudio_2do_Nv = $this->getEstudio_2do_Nvl();
    $Empleados_id_Empleado = $this->getEmpleados_id_Empleado();

    $sql = "INSERT IGNORE INTO `estudios`(`id_Estudio`, `Nivel_Acad`, `Titulo_Obt`, `Mencion`, `
        Estudio_2do_Nvl`, `Empleados_id_Empleado`)
    VALUES (
        NULL,
        '$Nivel_Acad',
        '$Titulo_Obt',
        '$Mencion',
        '$Estudio_2do_Nv',
        '$Empleados_id_Empleado'
    )";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setid_Estudio($conexion->insert_id);

    desconectar($conexion);
}
```

El método inicia una conexión a la base de datos, guarda el nivel académico, título académico, la mención, el estudio de segundo nivel y el id de empleado en variables locales. Luego prepara la instrucción SQL para insertar los valores, la ejecuta y muestra un mensaje de error en caso de fallar, finalmente guarda como atributo el id del registro realizado y cierra la conexión a base de datos.

editarEstudio:

```
public function editarEstudio() {  
    $conexion = conectar();  
  
    $Nivel_Acad = $this->getNivel_Acad();  
    $Titulo_Obt = $this->getTitulo_Obt();  
    $Mencion = $this->getMencion();  
    $Estudio_2do_Nv = $this->getEstudio_2do_Nvl();  
    $Empleados_id_Empleado = $this->getEmpleados_id_Empleado();  
  
    $sql = "UPDATE `estudios` SET  
        `Nivel_Acad`='$Nivel_Acad',  
        `Titulo_Obt`='$Titulo_Obt',  
        `Mencion`='$Mencion',  
        `Estudio_2do_Nvl`='$Estudio_2do_Nv'  
    WHERE  
        `Empleados_id_Empleado`='$Empleados_id_Empleado';  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    desconectar($conexion);  
}
```

El método inicia una conexión a la base de datos, guarda el nivel académico, título académico, la mención, el estudio de segundo nivel y el id de empleado en variables locales. Luego prepara la instrucción SQL para actualizar los valores, la ejecuta y muestra un mensaje de error en caso de fallar, finalmente cierra la conexión a base de datos.



Manual técnico del sistema.

consultar:

```
public function consultar($id_Empleado) {  
    $conexion = conectar();  
  
    $sql = "SELECT * FROM `estudios` WHERE `Empleados_id_Empleado`='$id_Empleado'";  
  
    $fetch = $conexion->query($sql);  
  
    $resultado = $fetch->fetch_assoc();  
  
    desconectar($conexion);  
  
    return $resultado;  
}
```

Inicialmente este método inicia una conexión a base de datos, almacena la consulta SQL en una variable, esta misma realiza una consulta que trae los datos de estudios de un empleado, el cual coincida con el id del empleado pasado como parámetro.

Se ejecutará la consulta y traerá los datos de estudios como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos de la base de datos.

informe.php

Clase padre:	No hereda
Nombre de la clase / interfaz:	Informe
Atributos:	
id_Informe	Privado. Id del informe médico en la base de datos.
Cert_Salud	Privado. Validez del certificado de salud.
Tarjeta_Vac	Privado. Validez de la tarjeta de vacunación.
Personas_id_Persona	Privado. Id de la persona a la que está asociado el informe médico.
Métodos:	
__construct	Constructor de la clase.
insertarInforme	Público. Registra un informe médico en la base de datos.
editarInforme	Público. Edita un informe médico en la base de datos.
consultar	Público. Retorna el informe médico de una persona.



Manual técnico del sistema.

Descripción de los métodos:

insertarInforme:

```
public function insertarInforme() {
    $conexion = conectar();

    $Cert_Salud = $this->getCert_Salud();
    $Tarjeta_Vac = $this->getTarjeta_Vac();
    $Personas_id_Persona = $this->getPersonas_id_Persona();

    $sql = "INSERT IGNORE INTO `informes`(`id_Informe`, `Cert_Salud`, `
        Tarjeta_Vac`, `Personas_id_Persona`) VALUES (
            NULL,
            '$Cert_Salud',
            '$Tarjeta_Vac',
            '$Personas_id_Persona'
        )";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setPersonas_id_Persona($conexion->insert_id);

    desconectar($conexion);
}
```

El método inicia una conexión a la base de datos, guarda los valores del certificado de salud y el de la tarjeta de vacunación. Luego prepara la instrucción SQL para insertar los valores, la ejecuta y muestra un mensaje de error en caso de fallar, finalmente guarda como atributo el id del registro realizado y cierra la conexión a base de datos.

editarInforme:

```
public function editarInforme() {
    $conexion = conectar();

    $Cert_Salud = $this->getCert_Salud();
    $Tarjeta_Vac = $this->getTarjeta_Vac();
    $Personas_id_Persona = $this->getPersonas_id_Persona();

    $sql = "UPDATE `informes` SET
    `Cert_Salud`='$Cert_Salud',
    `Tarjeta_Vac`='$Tarjeta_Vac'
    WHERE
    `Personas_id_Persona`='$Personas_id_Persona';";

    $conexion->query($sql) or die("error: ".$conexion->error);
    desconectar($conexion);
}
```

El método inicia una conexión a la base de datos guarda los valores del certificado de salud y el de la tarjeta de vacunación. Luego prepara la instrucción SQL para actualizar los valores, la ejecuta y muestra un mensaje de error en caso de fallar, finalmente cierra la conexión a base de datos.

consultar:

```
public function consultar($id_Persona) {
    $conexion = conectar();

    $sql = "SELECT * FROM `informes` WHERE `Personas_id_Persona`='$id_Persona';";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_assoc();

    desconectar($conexion);

    return $resultado;
}
```

Inicialmente este método inicia una conexión a base de datos, almacena la consulta SQL en una variable, esta misma realiza una consulta que trae los datos del informe médico de una persona, la cual coincida con el id de persona pasado como parámetro.



Manual técnico del sistema.

Se ejecutará la consulta y traerá los datos del informe médico como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos de la base de datos.

obrero.php

Clase padre:	Empleado, Persona
Nombre de la clase / interfaz:	Obrero
Atributos:	
id_Obrero	Privado. Id del obrero dentro de la base de datos.
Rol	Privado. Rol o función que cumple el obrero.
Métodos:	
__construct	Constructor de la clase.
insertarObrero	Público. Registra a un obrero en la base de datos.
editarObrero	Público. Edita los datos de un obrero en la base de datos.
mostrar	Público. Retorna todos los obreros que se hayan registrado.
consultar	Público. Retorna los datos de un obrero registrado



Manual técnico del sistema.

Descripción de los métodos.

insertarObrero:

```
public function insertarObrero() {  
  
    $this->insertarPersona();  
    $this->insertarEmpleado();  
  
    $conexion = conectar();  
  
    $Rol = $this->getRol();  
    $id_Empleado = $this->getid_Empleado();  
  
    $sql = "  
        INSERT INTO `obreros`(`id_Obrero`, `Rol`, `id_Empleado`)  
        VALUES(  
            NULL,  
            '$Rol',  
            '$id_Empleado'  
        );  
    ";  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    $this->setid_Obrero($conexion->insert_id);  
  
    desconectar($conexion);  
}
```

El método inicia ejecutando los métodos heredados insertarPersona e insertarEmpleado, luego inicia una conexión a base de datos y guarda el rol e id del empleado en variables locales, prepara la sentencia SQL para insertar. Ejecuta la consulta y lanza un mensaje de error en caso de fallar, almacena como atributo el id del obrero registrado y cierra la conexión a base de datos.



Manual técnico del sistema.

editarObrero:

```
public function editarObrero() {  
  
    $this->editarPersona();  
    $this->editarEmpleado();  
  
    $conexion = conectar();  
  
    $Rol = $this->getRol();  
    $id_Empleado = $this->getid_Empleado();  
  
    $sql = "  
        UPDATE  
        `obreros`  
        SET  
        `Rol`='$Rol'  
        WHERE  
        `id_Empleado`='$id_Empleado'  
    ";  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    desconectar($conexion);  
}
```

El método inicia ejecutando los métodos heredados editarPersona e editarEmpleado, luego inicia una conexión a base de datos y guarda el rol e id del empleado en variables locales, prepara la sentencia SQL para actualizar los valores. Ejecuta la consulta y lanza un mensaje de error en caso de fallar y cierra la conexión a base de datos.



Manual técnico del sistema.

mostrar:

```
public function mostrar(){
    $conexion = conectar();

    $sql = "
    SELECT * FROM `personas`,
    `empleados`, `obreros`
    WHERE
    `personas`.`id_Persona` = `empleados`.`id_Personas`
    AND
    `empleados`.`id_Empleado` = `obreros`.`id_Empleado`
    ;";

    $fetch = $conexion->query($sql);

    $resultados = $fetch->fetch_all(MYSQLI_ASSOC);

    $obreros = [];
    foreach ($resultados as $obrero) {
        $obreros[] = $obrero;
    }

    desconectar($conexion);

    return $obreros;
}
```

Este método usa polimorfismo, y se diferencia del método presente en su clase padre.

Inicia una conexión a base de datos y posteriormente guarda la instrucción SQL en una variable, este misma realiza una consulta que trae datos de personas, empleados y obreros, en los cuales se cumplan estas condiciones: el id de la persona debe coincidir con el id de persona almacenado en la tabla empleados, asimismo el id de empleado debe coincidir con el almacenado en la tabla obreros.

Luego, ejecuta la consulta para traer todos los registros que cumplen con las condiciones; estos mismos serán almacenados uno a uno en un arreglo. Cerrará la conexión y retornará la lista de empleados obreros.



Manual técnico del sistema.

consultar:

```
public function consultar($id_Persona) {
    $conexion = conectar();

    $sql = "
    SELECT * FROM `personas`,`empleados`,`obreros`
    WHERE
        `personas`.`id_Persona` = '$id_Persona'
    AND
        `empleados`.`id_Personas` = '$id_Persona'
    AND
        `empleados`.`id_Empleado` = `obreros`.`id_Empleado`
    ";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_assoc();

    desconectar($conexion);

    return $resultado;
}
```

El método inicia una conexión a base de datos, prepara la sentencia SQL usando el valor de su parámetro como criterio de búsqueda. Luego se ejecuta la orden, en caso de fallar mostrará un error.

Entonces, traerá los datos de la dirección como un arreglo asociativo. Al finalizar, cierra la conexión a base de datos y retorna los datos obtenidos.

persona.php

Clase padre:	No hereda
Nombre de la clase / interfaz:	Persona
Atributos:	
id_Personas	Privado. Id de la persona en la base de datos.
Nombre	Privado. Nombre de la persona.
Apellido	Privado. Apellido de la persona.
Cedula	Privado. Cédula de la persona.
Fecha_Nac	Privado. Fecha de nacimiento de la persona.
Ruta_Imagen	Privado. Ruta en el servidor a la imagen de la persona.
Métodos:	
__construct	Constructor de la clase.
insertarPersona	Público. Registra una persona en la base de datos.
editarPersona	Público. Edita los datos de una persona en la base de datos.
eliminarPersona	Público. Elimina a una persona de la base de datos.

Manual técnico del sistema.

Descripción de los métodos.

insertarPersona:

```
public function insertarPersona() {
    $conexion = conectar();

    $Nombre = $this->getNombre();
    $Apellido = $this->getApellido();
    $Cedula = $this->getCedula();
    $Fecha_Nac = $this->getFecha_Nac();
    $Ruta_Imagen = $this->getRuta_Imagen();
    $Sexo = $this->getSexo();

    $sql = "
        INSERT INTO `personas` (
            `id_Persona`,
            `Nombre`,
            `Apellido`,
            `Cedula`,
            `Fecha_Nac`,
            `Ruta_Imagen`,
            `Sexo`
        )
        VALUES(
            NULL,
            '$Nombre',
            '$Apellido',
            '$Cedula',
            '$Fecha_Nac',
            '$Ruta_Imagen',
            '$Sexo'
        );
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setid_Personas($conexion->insert_id);
    desconectar($conexion);
}
```

El método inicia una conexión a base de datos, luego almacena nombre, apellido, cédula, fecha de nacimiento, ruta en que almacena la imagen de la persona y su sexo en variables locales. Prepara la sentencia SQL para insertar, la ejecuta o muestra un mensaje de error en caso de fallar, finalmente, almacena como atributo el id de la persona registrada y cierra la conexión con la base de datos.

editarPersona:

```
public function editarPersona() {
    $conexion = conectar();

    $id_Personas = $this->getid_Personas();

    $Nombre = $this->getNombre();
    $Apellido = $this->getApellido();
    $Fecha_Nac = $this->getFecha_Nac();

    // La cédula no es actualizable
    $sql = "
        UPDATE
        `personas`
        SET
        `Nombre`='$Nombre',
        `Apellido`='$Apellido',
        `Fecha_Nac`='$Fecha_Nac'
        WHERE
        `id_Persona`='$id_Personas'
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    desconectar($conexion);
}
```

El método inicia una conexión a base de datos, luego almacena nombre, apellido, fecha de nacimiento e id de la persona en variables locales. Prepara la sentencia SQL para actualizar los valores, la ejecuta o muestra un mensaje de error en caso de fallar, finalmente cierra la conexión con la base de datos.



Manual técnico del sistema.

eliminarPersona:

```
public function eliminarPersona() {  
    $conexion = conectar();  
  
    $id_Personas = $this->getid_Personas();  
  
    // La cédula no es actualizable  
    $sql = "DELETE FROM `personas` WHERE `id_Persona`='$id_Personas'";  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    desconectar($conexion);  
}
```

El método inicia una conexión con la base de datos, almacena el id de la persona en una variable local, prepara la sentencia SQL para eliminar a la persona con ese id y la ejecuta, en caso de fallar mostrara un mensaje de error, para al final cerrar la conexión a base de datos.

Al eliminar a una persona se estarán eliminando los registros que estén asociados a esta en cascada.

telefono.php

Clase padre:	No hereda
Nombre de la clase / interfaz:	Telefono
Atributos:	
id_Telefono	Privado. Id del número telefónico en la base de datos
Prefijo	Privado. Prefijo telefónico.
Numero	Privado. Número telefónico.
Relacion	Privado. Relación de teléfono con la persona.
Métodos:	
__construct	Constructor de la clase.
insertarTelefono	Público. Inserta un teléfono en la base de datos.
editarTelefono	Público. Edita los datos de un teléfono en la base de datos.
consultar	Público. Retorna los teléfonos que pertenezcan a una ficha de contacto.
tipoTelefono	Público. Retorna la relación de un número de teléfono.
tipoTelefono	Público. Retorna la concatenación del prefijo y el número de teléfono.

Descripción de los métodos.

insertarTelefono:

```
public function insertarTelefono($Prefijo,$Numero,$Relacion,$id_Contacto) {
    $conexion = conectar();

    $sql = "
        INSERT IGNORE INTO `telefonos`
        (
            `id_Telefono`,
            `Prefijo`,
            `Numero`,
            `Relacion`,
            `id_Contacto`
        )
        VALUES
        (
            NULL,
            '$Prefijo',
            '$Numero',
            '$Relacion',
            '$id_Contacto'
        );
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    $this->setid_Telefono($conexion->insert_id);

    desconectar($conexion);
}
```

El método inicia una conexión a la base de datos y prepara la sentencia SQL para insertar usando como valores los parámetros recibidos, luego la ejecuta y en caso de fallar mostrará un mensaje de error, finalmente almacena como atributo el id del teléfono registrado y cierra la conexión con la base de datos.

editarTelefono:

```
public function editarTelefono($Prefijo,$Numero,$Relacion,$id_Contacto) {
    $conexion = conectar();

    $sql = "
        UPDATE
        `telefonos`
        SET
        `Prefijo`='$Prefijo',
        `Numero`='$Numero'
        WHERE
        `Relacion`='$Relacion'
        AND
        `id_Contacto`='$id_Contacto'
    ";

    $conexion->query($sql) or die("error: ".$conexion->error);
    desconectar($conexion);
}
```

El método inicia una conexión a la base de datos y prepara la sentencia SQL para editar usando como valores los parámetros recibidos, luego la ejecuta y en caso de fallar mostrará un mensaje de error, finalmente cierra la conexión con la base de datos.

consultar:

```
public function consultar($id_Contacto) {
    $conexion = conectar();

    $sql = "SELECT * FROM `telefonos` WHERE `id_Contacto` = '$id_Contacto'";
    $fetch = $conexion->query($sql);

    $resultados = $fetch->fetch_all(MYSQLI_ASSOC);

    $telefonos = [];
    foreach ($resultados as $telefono) {
        $telefonos[] = $telefono;
    }

    desconectar($conexion);

    return $telefonos;
}
```



Manual técnico del sistema.

El método inicia una conexión a la base de datos y prepara la sentencia SQL para consultar los números de teléfono pertenecientes a una ficha de contacto, luego la ejecuta y en caso de fallar mostrará un mensaje de error, finalmente cierra la conexión con la base de datos y retorna los registros obtenidos como un arreglo de arreglos asociativos.

tipoTelefono:

```
public function tipoTelefono($telefono) {  
    if ($telefono['Relacion'] == "P") {  
        $tipoTelefono = "Principal";  
    }  
    elseif ($telefono['Relacion'] == "S") {  
        $tipoTelefono = "Secundario";  
    }  
    elseif ($telefono['Relacion'] == "A") {  
        $tipoTelefono = "Auxiliar";  
    }  
    else {  
        $tipoTelefono = "Desconocido";  
    }  
    return $tipoTelefono;  
}
```

El método toma como parámetro un registro traído con el método consultar, verifica si la relación del teléfono es como teléfono principal, secundario o auxiliar, de no haber coincidencia con estas opciones lo retorna como desconocido.

tipoTelefono:

```
public function telefonoCompleto(){  
    return $this->Prefijo.$this->Numero;  
}
```

El método concatena el prefijo y el número telefónico y lo retorna.

usuario.php

Clase padre:	Persona
Nombre de la clase / interfaz:	Usuario
Atributos:	
id_Usuario	Privado. Id del usuario en la base de datos.
Rol	Privado. Rol o privilegios del usuario.
Contraseña	Privado. Contraseña de acceso del usuario.
Métodos:	
__construct	Constructor de la clase.
insertarUsuario	Público. Registra un usuario en la base de datos.
editarUsuario	Público. Edita los datos de un usuario.
chequeo_login	Público. Verifica que exista un usuario y contraseña específicos.
consultar	Público. Retorna los datos de un usuario.
mostrar	Público. Retorna todos los usuarios registrados.
contarUsuarios	Público. Retorna el número de usuarios registrados.



Manual técnico del sistema.

Descripción de los métodos.

insertarUsuario:

```
public function insertarUsuario() {  
    $conexion = conectar();  
  
    // METODO NO IMPLEMENTADO  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    $this->setid_Contacto($conexion->insert_id);  
  
    desconectar($conexion);  
}
```

El método no se encuentra implementado de una manera funcional por los momentos.

editarUsuario:

```
public function editarUsuario() {  
    $conexion = conectar();  
  
    // METODO NO IMPLEMENTADO  
  
    $conexion->query($sql) or die("error: ".$conexion->error);  
    desconectar($conexion);  
}
```

El método no se encuentra implementado de una manera funcional por los momentos.



Manual técnico del sistema.

chequeo_login:

```
public function chequeo_login() {
    $conexion = conectar();

    $cedula = $this->getCedula();
    $contraseña = $this->getContraseña();

    $sql = "SELECT * FROM personas, usuarios WHERE personas.id_Persona = usuarios.
        Personas_id_Persona and personas.cedula = '$cedula' AND usuarios.contraseña =
        '$contraseña'; ";

    $consulta_usuario = $conexion->query($sql) or die("error: ".$conexion->error);
    $usuario = $consulta_usuario->fetch_assoc();

    desconectar($conexion);

    return $usuario;
}
```

El método inicia una conexión a la base de datos, guarda la cedula y la contraseña del usuario en variables locales y prepara la sentencia SQL, esta misma busca la existencia de una persona con la cedula especificada que sea usuario y posea la contraseña especificada.

Cerrará la conexión y retornará el resultado obtenido o NULL en caso de no existir un usuario con esa combinación.

consultar:

```
public function consultar($id_Empleado) {
    $conexion = conectar();

    $sql = "SELECT * FROM `personas`,`usuarios` WHERE `id_Persona` and `
        Personas_id_Persona`;";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_assoc();

    desconectar($conexion);

    return $resultado;
}
```




Manual técnico del sistema.

mostrar:

```
public function mostrar() {
    $conexion = conectar();

    $sql = "SELECT * FROM `personas`,`usuarios` WHERE `personas`.`id_Persona` = `
        usuarios`.`Personas_id_Persona`";

    $fetch = $conexion->query($sql);

    $resultado = $fetch->fetch_all(MYSQLI_ASSOC);

    desconectar($conexion);

    return $resultado;
}
```

El método se conecta con la base de datos y prepara una sentencia SQL para consultar los usuarios existentes, la ejecuta o muestra un mensaje de error en caso de fallar, cierra la conexión y retorna los resultados obtenidos.

contarUsuarios

```
public function contarUsuarios() {
    $conexion = conectar();

    $sql = "SELECT COUNT(*) FROM `personas`,`usuarios` WHERE `personas`.`id_Persona`
        = `usuarios`.`Personas_id_Persona`";

    $fetch = $conexion->query($sql);

    $resultados = $fetch->fetch_all(MYSQLI_ASSOC);

    $conteo = $resultados[0]["COUNT(*)"];

    desconectar($conexion);

    return $conteo;
}
```



Manual técnico del sistema.

El método se conecta con la base de datos y prepara una sentencia SQL para consultar el número de usuarios existentes, la ejecuta o muestra un mensaje de error en caso de fallar, cierra la conexión y retorna el conteo de usuarios registrados.



Manual técnico del sistema.

Controladores.

base-datos.php

Limpia la base de datos y carga datos nuevos de prueba.

```
1 <?php
2 require('conexion.php');
3 $con = conectar();
4
5 // Vacía la base de datos
6 $sql = 'DELETE FROM "personas"';
7 $con->query($sql);
8
9 // Lista de archivos sql a importar
10 // Nota: deben importarse de manera ordenada
11 $archivos = [
12     'personas',
13     'empleados',
14     'obreros',
15     'docentes',
16     'administrativos',
17     'contactos',
18     'telefonos',
19     'direcciones',
20     'estudios',
21     'carga-horaria',
22     'informes',
23     'usuarios',
24 ];
25
26 // Itera los archivos y los va ejecutando secuencialmente
27 foreach ($archivos as $sql) {
28     $instrucciones = file_get_contents("../sql/".$sql.".sql");
29     $con->query($instrucciones);
30 }
31
32 desconectar($con);
33
34 header('Location: ../lobby/index.php?BD_RESTAURADA');
35 ?>
```

Este controlador empieza incluyendo el controlador conexión.php e instanciando una conexión con la base de datos, luego de eso, vaciará la tabla personas (tablas hijas serán eliminadas en cascada). Se contará con un array con los nombres (sin incluir la extensión) de cada archivo sql a importar. Los nombres de los archivos sql incluidos por defecto son los siguientes:

```
1 administrativos.sql
2 carga-horaria.sql
3 contactos.sql
4 direcciones.sql
5 docentes.sql
6 empleados.sql
7 estudios.sql
8 informes.sql
9 obreros.sql
10 personas.sql
11 rellenar-bd.sql
12 telefonos.sql
13 usuarios.sql
```



Manual técnico del sistema.

Entonces entrará en un bucle foreach en que usará la función *file_get_contents* para obtener el contenido de un archivo sql como un string que posteriormente será ejecutado con el método *query* de la conexión instanciada con la base de datos. Hará esto para cada archivo mencionado anteriormente.

Finalmente, cerrará la conexión a la base de datos y redirigirá al usuario (administrador) a menú principal del lobby, a la vez de enviar una variable GET por la URL para mostrar un mensaje de operación exitosa.



Manual técnico del sistema.

conexion.php

Gestiona la conexión y desconexión con la base de datos.

Este controlador posee tres funciones, *conectar*, *comprobar_bdy* y *desconectar*.

```
function conectar() {  
  
    $servidor    = "localhost";  
    $usuario     = "root";  
    $bd          = "ingsoft_ii";  
    $clave       = "";  
  
    //instancia la clase mysqli y se conecta a la base de datos  
    $conexion = new mysqli($servidor,$usuario,$clave,$bd);  
  
    $conexion->set_charset("utf8");  
  
    if($conexion->connect_errno) {  
        echo "Error al conectar ". $this->conexion->connect_errno."<br>";  
        exit();  
    }  
  
    return $conexion;  
}
```

La función conectar cuenta con las variables para instanciar una conexión a base de datos mediante mysqli, se debe especifica el servidor, usuario a usar, el nombre de la base de datos (por defecto ingsoft_ii) y la contraseña de usuario.

Al instanciar la conexión se establece el juego de caracteres a utf-8, así como también se verifica que no haya un error de conexión, que en caso de existir, mostrará el error y detendrá la ejecución.

En caso de instanciarse con éxito se retornará la conexión como un objeto de mysqli.

Manual técnico del sistema.

```
function comprobar_bd() {  
    $servidor = "localhost";  
    $usuario  = "root";  
    $clave    = "";  
  
    //instancia la clase mysqli y se conecta a la base de datos  
    $conexion = new mysqli($servidor,$usuario,$clave);  
  
    $conexion->set_charset("utf8");  
  
    // Verifica si la base de datos existe  
    $sql = "show DATABASES LIKE 'ingsoft_ii'";  
    $res = $conexion->query($sql);  
    if ($res->num_rows >= 1) {  
        return true;  
    }  
    else {  
        return false;  
    }  
}
```

La segunda función, comprobar_bd, se conecta a base de datos como en la función anterior, con la diferencia de que no se especifica una base de datos. Posterior a eso, consulta si existe la base de datos, si hay al menos una fila (base de datos que coincida) retorna true, por lo contrario retornará false.

```
function desconectar($conexion) {  
    if ($conexion) {  
        $conexion->close();  
    }  
}
```

Por último, está la función desconectar. Toma como parámetro la conexión generada por la función conectar. Si esa conexión existe, la cierra.



Manual técnico del sistema.

control-empleados.php

Evalúa los datos enviados desde los formularios y utiliza clases para realizar acciones en la base de datos (Crear, editar, consultar y eliminar) según el formulario que se llene.

```
session_start();

if (!$SESSION['login']) {
    header('Location: ../index.php');
    exit();
}
```

En primer lugar, el usuario debe haber iniciado sesión para acceder al controlador.

```
require('conexion.php');

function subirImagen($cedula) {
    // Código adaptado de https://www.w3schools.com/php/php_file_upload.asp
    $direct_objetivo = "../img/subidas/";

    // Se asigna la cédula de la persona como nombre de archivo. Se conserva la extensión
    $t = explode('.', $_FILES["foto"]["name"]);
    $nombre_archivo = $cedula.".".$t[1];

    $archivo_objetivo = $direct_objetivo . $nombre_archivo;
    $estado_subida = 1;
    $filetype_imagen = strtolower(pathinfo($archivo_objetivo, PATHINFO_EXTENSION));

    // Verifica si la imagen enviada existe
    $check = getimagesize($_FILES["foto"]["tmp_name"]);

    if($check !== false) {$estado_subida = 1;}
    else {$estado_subida = 0;}

    // Verifica si el archivo ya existe
    if (file_exists($archivo_objetivo)) {
        unlink($archivo_objetivo);
    }

    // Validación de tamaño del archivo subido
    if ($_FILES["foto"]["size"] > 500000) {
        $estado_subida = 0;
    }

    // Limitar los formatos elegibles de JPG, JPEG, PNG y GIF
    if ($filetype_imagen != "jpg" && $filetype_imagen != "png" &&
        $filetype_imagen != "jpeg" && $filetype_imagen != "gif" ) {
        $estado_subida = 0;
    }

    // Verifica si no hay errores
    if ($estado_subida == 0) {
        // Establecer nula la imagen en BD
    }
    // Intenta subir la imagen
    else {
        if (move_uploaded_file($_FILES["foto"]["tmp_name"], $archivo_objetivo)) {
            return $nombre_archivo;
        }
        else {
            // Establecer nula la imagen en BD
        }
    }
}
```

En segundo lugar, este controlador cuenta con una función llamada *subirImagen* la cual recibe la cédula del empleado en cuestión.



Manual técnico del sistema.

Se establece el directorio al que va a ser enviada la imagen, luego se toman tanto el nombre como la extensión del archivo, verifica que haya recibido una imagen desde el formulario,

Luego de esto, verifica si ya existe una imagen previa, de existir la elimina.

Verifica el tamaño de la imagen a subir y finalmente la sube al directorio especificado usando la cédula del empleado como nombre y conservando su tipo de archivo (jpg, png, jpeg o gif).

```
// Condicionales para distinguir acciones
if
(
    (
        isset($_POST['paso-1'],$_POST['orden'])
        or
        isset($_POST['paso-2'],$_POST['orden'])
        or
        isset($_POST['editar'],$_POST['orden'])
        or
        isset($_POST['eliminar'],$_POST['orden'])
    )
    and
    !empty($_POST['orden'])
)
{
    ...
}
```

Este controlador cuenta con un condicional que utiliza las variables enviadas a la hora de enviar un formulario. Con el propósito de validar que se envió correctamente el formulario.

Así como verificar que se haya enviado una orden a realizar.

Si los datos recibidos cumplen con estas condiciones, verifica que tipo de orden recibió, pudiendo ser insertar, editar o eliminar.

```
else {
    header('Location:../lobby/index.php');
}
```

En dado caso que no se envíe ninguna instrucción de un formulario, se enviara al usuario al menú principal del lobby

Manual técnico del sistema.

Caso Insertar:

```
// ORDENES PARA INSERTAR
if ($_POST['orden'] == 'insertar') {
    switch ($_POST['T_empleado']) {
        // Caso Obrero
        case 0:
            require("../clases/obrero.php");
            $emp = new Obrero();

            // Persona
            $emp->setNombre($_POST['nombre']);
            $emp->setApellido($_POST['apellido']);
            $emp->setCedula($_POST['cedula']);
            $emp->setFecha_Nac($_POST['F_nac']);
            $emp->setSexo($_POST['sexo']);

            // Foto
```

Para el caso de insertar, se tomará en cuenta mediante un switch que tipo de empleado se va a registrar, pudiendo ser obrero, docente o administrativo.

```
        // Caso Obrero
        case 0:
            require("../clases/obrero.php");
            $emp = new Obrero();

            // Persona
            $emp->setNombre($_POST['nombre']);
            $emp->setApellido($_POST['apellido']);
            $emp->setCedula($_POST['cedula']);
            $emp->setFecha_Nac($_POST['F_nac']);
            $emp->setSexo($_POST['sexo']);

            // Foto
            $foto = subirImagen($_POST['cedula']);
            $emp->setRuta_Imagen($foto);

            // Empleado
            $emp->setFecha_Ingreso($_POST['FI_nomina']);

            // Obrero
            $emp->setRol($_POST['rol']);

            $emp->insertarObrero();
            break;
```

Para el caso de obreros, se instancia un objeto de la clase obrero al que se le asignan sus datos personales, se usa la función subirImagen y se establece la ruta de la imagen subida, luego se agregará la fecha de ingreso a nómina y el rol de obrero que cumple, finalmente se ejecutará el método insertarObrero para registrarlo en la base de datos y saldrá del switch.

```
// Caso Docente
case 1:
    require("../clases/docente.php");
    $emp = new Docente();

    // Persona
    $emp->setNombre($_POST['nombre']);
    $emp->setApellido($_POST['apellido']);
    $emp->setCedula($_POST['cedula']);
    $emp->setFecha_Nac($_POST['F_nac']);
    $emp->setSexo($_POST['sexo']);

    // Foto
    $foto = subirImagen($_POST['cedula']);
    $emp->setRuta_Imagen($foto);

    // Empleado
    $emp->setFecha_Ingreso($_POST['FI_nomina']);

    // Docente
    $emp->setHoras_Clase_S($_POST['horas_acad']);
    $emp->setArea($_POST['area']);

    $emp->insertarDocente();
    break;
```

En el caso de ser docente, funcionará de modo similar, instanciará un objeto de la clase Docente, establecerá los datos personales, la imagen y la fecha de ingreso a nómina; se diferencia del anterior dado que se establece el número de horas de clase semanales y el área al que corresponde, finalmente se ejecuta insertarDocente para anexarlo a la base de datos.

```
// Caso Administrativo
case 2:
    require("../clases/administrativo.php");
    $emp = new Administrativo();

    // Persona
    $emp->setNombre($_POST['nombre']);
    $emp->setApellido($_POST['apellido']);
    $emp->setCedula($_POST['cedula']);
    $emp->setFecha_Nac($_POST['F_nac']);
    $emp->setSexo($_POST['sexo']);

    // Foto
    $foto = subirImagen($_POST['cedula']);
    $emp->setRuta_Imagen($foto);

    // Empleado
    $emp->setFecha_Ingreso($_POST['FI_nomina']);

    // Administrativo (Aún no tiene datos específicos)

    $emp->insertarAdministrativo();
    break;
```

En caso de ser un empleado administrativo, se instancia un objeto de la clase Administrativo, se establecen los datos personales, la imagen y la fecha de ingreso a nómina, por los momentos

Manual técnico del sistema.

el personal administrativo no cuenta con atributos exclusivos. Finalmente se ejecuta insertarAdministrativo para registrarlo en base de datos.

```
// Respuesta en caso aparte
default:
    header('Location: ../lobby/consultar/index.php?error');
    break;
}
```

En caso de no cumplir ninguno de los casos anteriores, redireccionará por defecto al área de consulta y enviando una variable GET.

```
// Ficha de contacto
require('../clases/contacto.php');
$con = new Contacto();

$con->setCorreo($_POST['correo']);
$con->setid_Personas($emp->getid_Personas());

$con->insertarContacto();

// Direccion
require('../clases/direccion.php');
$dir = new Direccion();

$dir->setMunicipio($_POST['municipio']);
$dir->setParroquia($_POST['parroquia']);
$dir->setDireccion($_POST['direccion']);
$dir->setContactos_id_Contacto($con->getid_Contacto());

$dir->insertarDireccion();

// Telefonos
require('../clases/telefono.php');
$tel = new Telefono();

// Telefonos principal, secundario y auxiliar.
// En ese mismo orden
$tel->insertarTelefono($_POST['pref_P'], $_POST['tel_P'], 'P', $con->getid_Contacto());
$tel->insertarTelefono($_POST['pref_S'], $_POST['tel_S'], 'S', $con->getid_Contacto());
$tel->insertarTelefono($_POST['pref_A'], $_POST['tel_A'], 'A', $con->getid_Contacto());

// Informe
require('../clases/informe.php');
$inf = new Informe();

// Si esta vacunado con al menos una dosis
if ((!empty($_POST['vacuna']) and !empty($_POST['dosis_vacuna'])) and ($_POST['dosis_vacuna'] >=1)) {
    $inf->setCert_Salud("Valido");
    $inf->setTarjeta_Vac("Valido");
}
// Si no lo esta
else {
    $inf->setCert_Salud("No valido");
    $inf->setTarjeta_Vac("No valido");
}

$inf->setPersonas_id_Persona($emp->getid_Personas());

$inf->insertarInforme();
```

Una vez registrados los datos personales y laborales del empleado, se instanciara las clases Contacto, Telefono e Informe.

En el caso de Contacto, se almacena el correo electrónico y el id de la persona, para luego insertar en la base de datos.

Manual técnico del sistema.

En el caso de Telefono, se almacena el prefijo telefónico, número de teléfono, relación del teléfono (Principal, secundario o auxiliar) y el id de la persona, para luego insertar en la base de datos.

Luego para Informe, se verifica el que el empleado tenga al menos una vacuna aplicada y que su certificado de salud sea válido; en caso de que ambos sean válidos, se establecen válidos, en caso contrario, ambos se establecerán como no válidos, luego de eso, se anexará esta información a la base de datos.

```
// Estudios
require('../clases/estudio.php');
$est = new Estudio();

$est->setNivel_Acad($_POST['N_academico']);

// Si el empleado tiene estudios universitarios
if ($_POST['N_academico'] >= 3) {
    $est->setTitulo_Obt($_POST['titulo']);
    $est->setMencion($_POST['mencion']);
    $est->setEstudio_2do_Nvl($_POST['E_2do_nivel']);
}
$est->setEmpleados_id_Empleado($emp->getId_Empleado());

$est->insertarEstudio();

// Carga Horaria
require('../clases/carga-horaria.php');
$ch = new Carga_Horaria();

$ch->setCarga_Horaria_Semanal($_POST['horas']);
$ch->setEmpleados_id_Empleado($emp->getId_Empleado());

$ch->insertarCarga_Horaria();

header('Location: ../lobby/consultar/index.php');
```

Posteriormente se instancian la clase Estudio y Carga_Horaria.

Para estudios, se almacena el nivel académico, en caso de que sea mayor o igual a 3 (Nivel universitario) se almacena el título que posea, la mención y estudios de segundo nivel en caso de tener, por último, el id del empleado y se anexa a base de datos.

Para Carga_Horaria, se almacena el número de horas laborales por semana que cumple el empleado, y el id del empleado para luego anexarlo a base de datos.

Manual técnico del sistema.

Caso editar:

Para el caso de editar, el procedimiento es similar al de insertar, con las diferencias de que al instanciar las clases Obrero, Docente o Administrativo, no se especifica ni número de cédula ni la imagen, estos mismos no son editables.

Caso Obrero	Caso Docente
<pre> switch (\$_POST['T_empleado']) { // Caso Obrero case 1: require("../clases/obrero.php"); \$emp = new Obrero(); // Persona \$emp->setid_Personas(\$_POST['id_Persona']); \$emp->setNombre(\$_POST['nombre']); \$emp->setApellido(\$_POST['apellido']); // \$emp->setCedula(\$_POST['cedula']); \$emp->setFecha_Nac(\$_POST['F_nac']); \$emp->setSexo(\$_POST['sexo']); // // Foto // \$foto = subirImagen(\$_POST['cedula']); // \$emp->setRuta_Imagen(\$foto); // Empleado \$emp->setFecha_Ingreso(\$_POST['FI_nomina']); // Obrero \$emp->setRol(\$_POST['rol']); \$emp->editarObrero(); break; </pre>	<pre> // Caso Docente case 2: require("../clases/docente.php"); \$emp = new Docente(); // Persona \$emp->setid_Personas(\$_POST['id_Persona']); \$emp->setNombre(\$_POST['nombre']); \$emp->setApellido(\$_POST['apellido']); // \$emp->setCedula(\$_POST['cedula']); \$emp->setFecha_Nac(\$_POST['F_nac']); \$emp->setSexo(\$_POST['sexo']); // // Foto // \$foto = subirImagen(\$_POST['cedula']); // \$emp->setRuta_Imagen(\$foto); // Empleado \$emp->setFecha_Ingreso(\$_POST['FI_nomina']); // Docente \$emp->setHoras_Clase_S(\$_POST['horas_acad']); \$emp->setArea(\$_POST['area']); \$emp->editarDocente(); break; </pre>
Caso Administrativo	Caso aparte
<pre> // Caso Administrativo case 3: require("../clases/administrativo.php"); \$emp = new Administrativo(); // Persona \$emp->setid_Personas(\$_POST['id_Persona']); \$emp->setNombre(\$_POST['nombre']); \$emp->setApellido(\$_POST['apellido']); // \$emp->setCedula(\$_POST['cedula']); \$emp->setFecha_Nac(\$_POST['F_nac']); \$emp->setSexo(\$_POST['sexo']); // // Foto // \$foto = subirImagen(\$_POST['cedula']); // \$emp->setRuta_Imagen(\$foto); // Empleado \$emp->setFecha_Ingreso(\$_POST['FI_nomina']); // Administrativo (Aún no tiene datos específicos) \$emp->editarAdministrativo(); break; </pre>	<pre> // Respuesta en caso aparte default: header('Location: ../lobby/consultar/index.php?error'); break; </pre>

Manual técnico del sistema.

Una vez recibidos los datos nuevos del empleado, se envían a la base de datos con los métodos *editarObrero*, *editarDocente* y *editarAdministrativo* de sus respectivos casos.

```
// Ficha de contacto
require('../clases/contacto.php');
$con = new Contacto();

$con->setCorreo($_POST['correo']);
$con->setid_Personas($emp->getid_Personas());

$con->editarContacto();

// Direccion
require('../clases/direccion.php');
$dir = new Direccion();

$dir->setMunicipio($_POST['municipio']);
$dir->setParroquia($_POST['parroquia']);
$dir->setDireccion($_POST['direccion']);
$dir->setContactos_id_Contacto($con->getid_Contacto());

$dir->editarDireccion();

// Telefonos
require('../clases/telefono.php');
$tel = new Telefono();

// Telefonos principal, secundario y auxiliar.
// En ese mismo orden
$tel->editarTelefono($_POST['pref_P'], $_POST['tel_P'], 'P', $con->getid_Contacto());
$tel->editarTelefono($_POST['pref_S'], $_POST['tel_S'], 'S', $con->getid_Contacto());
$tel->editarTelefono($_POST['pref_A'], $_POST['tel_A'], 'A', $con->getid_Contacto());
```

Posteriormente continua con las clases Contacto, Dirección y Telefono. Establece sus datos y se envían a base de datos con sus respectivos métodos para editar.

```
// Infome
require('../clases/informe.php');
$inf = new Informe();

// Si esta vacunado con al menos una dosis
if ((!empty($_POST['vacuna']) and !empty($_POST['dosis_vacuna'])) and ($_POST['dosis_vacuna'] >=1)) {
    $inf->setCert_Salud("Valido");
    $inf->setTarjeta_Vac("Valido");
}
// Si no lo esta
else {
    $inf->setCert_Salud("No valido");
    $inf->setTarjeta_Vac("No valido");
}

$inf->setPersonas_id_Persona($emp->getid_Personas());

$inf->editarInforme();

// Estudios
require('../clases/estudio.php');
$est = new Estudio();

$est->setNivel_Acad($_POST['N_academico']);

// Si el empleado tiene estudios universitarios
if ($_POST['N_academico'] >= 3) {
    $est->setTitulo_Obt($_POST['titulo']);
    $est->setMencion($_POST['mencion']);
    $est->setEstudio_2do_Nvl($_POST['E_2do_nivel']);
}
$est->setEmpleados_id_Empleado($emp->getid_Empleado());

$est->editarEstudio();

// Carga Horaria
require('../clases/carga-horaria.php');
$sch = new Carga_Horaria();

$sch->setCarga_Horaria_Semanal($_POST['horas']);
$sch->setEmpleados_id_Empleado($emp->getid_Empleado());

$sch->editarCarga_Horaria();

header('Location: ../lobby/consultar/index.php');
}
```

Se sigue el mismo procedimiento para las clases Informe, Estudio y Carga_Horaria. Finalmente se envía al usuario al área de consulta.



Manual técnico del sistema.

Caso eliminar:

```
// ORDENES PARA ELIMINAR
elseif ($_POST['orden'] == 'eliminar') {
    require('../clases/persona.php');
    $per = new Persona();
    $per->setid_Personas($_POST['id_Persona']);
    $per->eliminarPersona();
    header('Location:../lobby/consultar/index.php');
}
```

Para el caso eliminar, se instancia la clase persona, se asigna el id de la persona (empleado) y se ejecuta el método eliminarPersona para eliminar su registro de la base de datos.

Posteriormente el usuario es enviado al área de consulta.

Manual técnico del sistema.

docente-excel.php

Lista los docentes existentes en la base de datos y los exporta en formato Excel.

```
session_start();

if (!$SESSION['login']) {
    header('Location: ../index.php');
    exit();
}
```

En un comienzo el controlador verifica que el usuario haya iniciado sesión.

```
require('conexion.php');
require('../clases/docente.php');
require('../clases/calculos.php');

// Consulta de todos los docentes
$doc = new Docente();
$calc = new Calculo();

$lista_doc = $doc->mostrar();

$archivoExcel = "Reporte de docentes";

//header info for browser
header("Content-Type: application/xls");
header("Content-Disposition: attachment; filename=$archivoExcel.xls");
header("Pragma: no-cache");
header("Expires: 0");

/*****Start of Formatting for Excel*****/
//define separator (defines columns in excel & tabs in word)

//File Name
$separador = "\t"; //tabulacion
$salto = $salto; //salto de linea

// Columnas de título
```

Luego de eso, llamará el controlador de conexión a base de datos, la clase Docente y la interfaz Calculo, instanciará estas dos últimas. Luego se creará la variable lista_doc, la cual contiene el listado general de docentes registrados con el método mostrar de la clase Docente.

Se especificará contenido mediante header al navegador, indicando que es un archivo Excel adjunto.



Manual técnico del sistema.

Luego se especifica el nombre del archivo y las variables que especificarán saltos de línea (paso a siguiente fila enfocando la primera columna) y tabulaciones (paso a la siguiente columna a la derecha) al generar el Excel.

```
// Columnas de titulo
echo utf8_decode("Nombre").$separador;
echo utf8_decode("Apellido").$separador;
echo utf8_decode("Cédula").$separador;
echo utf8_decode("Fecha de nacimiento").$separador;
echo utf8_decode("Horas académicas").$separador;
echo utf8_decode("Tiempo en nomina<").$separador;
echo utf8_decode("Área").$separador;

print($salto);

// Contenido de la consulta
foreach ($lista_doc as $Docente) {
    echo utf8_decode($Docente['Nombre']).$separador;
    echo utf8_decode($Docente['Apellido']).$separador;
    echo utf8_decode($Docente['Cedula']).$separador;
    echo utf8_decode($Docente['Fecha_Nac']).$separador;
    echo utf8_decode($Docente['Horas_Clase_S']. " Horas semanales").$separador;
    echo utf8_decode($calc->diferenciaF($Docente['Fecha_Ingreso'])).$separador;
    echo utf8_decode($Docente['Area']).$separador;
    print($salto);
}
```

Una vez definido esto, se comienza a formatear el documento Excel al usar echo. Primero la cabecera usando la función utf8_decode para convertir las cadenas de texto a codificación ISO-8859-1.

Luego, hará un bucle foreach que itere cada resultado de la lista de docentes registrados, e imprimirá los atributos: Nombre, Apellido, Cédula, Fecha de nacimiento, el número de horas de clase que imparte por semana, realizará un cálculo para mostrar cuánto tiempo lleva en nómina a partir de su fecha de ingreso, por último, imprimirá el área del docente y hará un salto de línea para repetir el proceso con el siguiente registro hasta que no hayan más.

Finalmente el archivo Excel será descargado por el navegador del usuario.

Manual técnico del sistema.



Consultar docentes

localhost:8080/sistema_ing_soft_II/sistema/lobby/consultar/index.php

SISNO-18 Personal

Área de consulta

Selector de consulta

Consultar Personal Consultar Obreros Consultar Docentes Consultar Administrativos Consultar Usuarios

Consulta de docentes.

Registrar empleado Descargar Excel Descargar PDF

Nombre	Apellido	Cédula	Fecha de nacimiento	Horas académicas	Tiempo en nomina	Área	Acciones
Terri	Fist	V93909125	2022-10-18	16 Horas semanales	5 Meses	Inglés	consultar
Ramsay	Roussel	V69823294	2022-07-21	26 Horas semanales	5 Meses	Castellano	consultar
Frannie	Pirkis	V10113385	2022-08-06	18 Horas semanales	5 Meses	Matemática	consultar

Sistema de nómina

Hecho con tailwindcss por Elber Rondón

Reporte de docentes(4).xls - Microsoft Excel

ARCHIVO INICIO INSERTAR DISEÑO DE PÁGINA FÓRMULAS DATOS REVISAR VISTA Iniciar sesión

Calibri 11

Pegar Fuente Alineación Número Estilos

Formato condicional Dar formato como tabla Estilos de celda Celdas Modificar

E7 32 Horas semanales

	A	B	C	D	E	F	G	H	I
1	Nombre	Apellido	Cédula	Fecha de nac	Horas acadé	Tiempo en n	Área		
2	Terri	Fist	V93909125	18-10-2022	16 Horas sen	5 Meses	Inglés		
3	Ramsay	Roussel	V69823294	21-07-2022	26 Horas sen	5 Meses	Castellano		
4	Frannie	Pirkis	V10113385	06-08-2022	18 Horas sen	5 Meses	Matemática		
5	Evaleen	Bindley	V32896787	06-04-2022	13 Horas sen	1 Año	Cálculo		
6	Reinaldos	Brabham	V35029573	01-09-2022	10 Horas sen	5 Meses	Educ. Física		
7	Amy	Liverseegee	V66509818	01-01-2022	32 Horas sen	9 Meses	Química		
8									
9									
10									
11									
12									
13									

Reporte de docentes(4)

LISTO 100 %



Manual técnico del sistema.

docente-pdf.php

Lista los docentes existentes en la base de datos y los exporta en formato PDF.

```
<?php
session_start();

if (!$SESSION['login']) {
    header('Location: ../index.php');
    exit();
}
```

Inicialmente se verifica que la sesión del usuario exista.

```
require('conexion.php');
require('../clases/docente.php');
require('../clases/calculos.php');

require('../fpdf185/fpdf.php');

// Consulta de todos los docentes
$doc = new Docente();

$calc = new Calculo();

$lista_doc = $doc->mostrar();

$pdf = new FPDF();
```

Se llama el controlador de conexión a base de datos, la clase Docente y la interfaz Calculo, además de archivo principal de fpdf.

Se instancian la clase Docente y la interfaz Calculo, se usa el método mostrar de Docente para obtener la lista de docentes registrados. Luego se instancia la clase FPDF.

```
$pdf->SetFont('Arial','',14);
$pdf->AddPage('L','Legal');

$pdf->SetTitle('Reporte de docentes');

$pdf->Write(7,'Docentes registrados en el sistema');
$pdf->Ln(20);
```



Manual técnico del sistema.

Se establece la tipografía como Arial de 14 puntos sin estilos adicionales; se establece el tamaño del documento en tamaño Legal (Oficio) y orientación Landscape (vertical).

Luego se define el nombre del documento, se inserta el título del documento y un salto de línea.

```
$pdf->SetFont('Arial','B',10);  
// Cabecera  
$pdf->Cell(25,7,utf8_decode("Nombre"),1);  
$pdf->Cell(25,7,utf8_decode("Apellido"),1);  
$pdf->Cell(25,7,utf8_decode("Cédula"),1);  
$pdf->Cell(40,7,utf8_decode("Fecha de nacimiento"),1);  
$pdf->Cell(60,7,utf8_decode("Horas académicas"),1);  
$pdf->Cell(60,7,utf8_decode("Tiempo en nomina"),1);  
$pdf->Cell(40,7,utf8_decode("Área"),1);  
$pdf->Ln();  
  
$pdf->SetFont('Arial','',10);  
foreach ($lista_doc as $Docente) {  
    $pdf->Cell(25,7,utf8_decode($Docente['Nombre']),1);  
    $pdf->Cell(25,7,utf8_decode($Docente['Apellido']),1);  
    $pdf->Cell(25,7,utf8_decode($Docente['Cedula']),1);  
    $pdf->Cell(40,7,utf8_decode($Docente['Fecha_Nac']),1);  
    $pdf->Cell(60,7,utf8_decode($Docente['Horas_Clase_S']." Horas semanales"),1);  
    $pdf->Cell(60,7,utf8_decode($calc->diferenciaF($Docente['Fecha_Ingreso'])),1);  
    $pdf->Cell(40,7,utf8_decode($Docente['Area']),1);  
    $pdf->Ln();  
}  
  
$pdf->Output();
```

Se establece la tipografía como Arial de 10 puntos en negrita, luego se imprimen celdas de tamaño fijo y con borde y un salto de línea.

Luego se retira la propiedad de negrita a la tipografía, y se entra en un bucle foreach y se imprime la información de los docentes en celdas separadas, en el caso de tiempo en nómina, se calcula en base a la fecha actual y la fecha de ingreso.

Finalmente se imprime el documento completo para que el usuario lo pueda visualizar o descargar (dependiendo de la configuración del navegador).

Manual técnico del sistema.



Consultar docentes

Reporte de docentes - doc.pdf

localhost:8080/sistema_ing_soft_II/sistema/lobby/consultar/index.php

SISNO-18

Personal

Área de consulta

Selector de consulta

Consultar Personal Consultar Obreros Consultar Docentes Consultar Administrativos Consultar Usuarios

Consulta de docentes.

Registrar empleado Descargar Excel Descargar PDF

Nombre	Apellido	Cédula	Fecha de nacimiento	Horas académicas	Tiempo en nomina	Área	Acciones
Terri	Fist	V93909125	2022-10-18	16 Horas semanales	5 Meses	Inglés	consultar
Ramsay	Roussel	V69823294	2022-07-21	26 Horas semanales	5 Meses	Castellano	consultar
Frannie	Pirkis	V10113385	2022-08-06	18 Horas semanales	5 Meses	Matemática	consultar

Sistema de nómina

Hecho con tailwindcss por Elber Rondón

Consultar docentes

Reporte de docentes - doc.pdf

localhost:8080/sistema_ing_soft_II/sistema/controladores/docente-pc

1 de 1

Tamaño automático

Docentes registrados en el sistema

Nombre	Apellido	Cédula	Fecha de nacimiento	Horas académicas	Tiempo en nomina	Área
Terri	Fist	V93909125	2022-10-18	16 Horas semanales	5 Meses	Inglés
Ramsay	Roussel	V69823294	2022-07-21	26 Horas semanales	5 Meses	Castellano
Frannie	Pirkis	V10113385	2022-08-06	18 Horas semanales	5 Meses	Matemática
Evaleen	Bindley	V32896787	2022-04-06	13 Horas semanales	1 Año	Cálculo
Reinaldos	Brabham	V35029573	2022-09-01	10 Horas semanales	5 Meses	Educ. Física
Amy	Liverseege	V85009818	2022-01-01	32 Horas semanales	9 Meses	Química

Manual técnico del sistema.

login.php

Verifica los datos de ingreso del usuario (Cédula y contraseña), en caso de ser correcto, inicia una sesión en el sistema y concede el acceso al usuario.

```
// Se llama conexion
require("conexion.php");

// Llama las clases
require_once('../clases/persona.php');
require_once('../clases/usuario.php');

$usuarios = new usuario();
```

Inicialmente se llama el controlador de conexión a base de datos y los archivos que incluyen las clases Persona y Usuario.

```
// Si los campos de cedula y contraseña fueron llenados
if (isset($_POST['cedula'],$_POST['contraseña'])) {

    // Verifica que la cadena recibida no esté vacía
    if (!empty($_POST['cedula']) and !empty($_POST['contraseña'])) {

        $cedula = $_POST['nacionalidad'] . $_POST['cedula'];
        $contraseña = $_POST['contraseña'];

        $usuarios->setCedula($cedula);
        $usuarios->setContraseña($contraseña);

        $chequeo_login = $usuarios->chequeo_login();

        // verifica que la base de datos no dé una respuesta nula
        if ($chequeo_login) {
            session_start();

            $_SESSION['datos_login'] = $chequeo_login;
            $_SESSION['login'] = true;

            header('Location: ../lobby/index.php');
        }

        // en caso de retornar nulo regresa al menú
        else {
            // devuelve al menú por error de datos (usuario o clave errados)
            header('Location: ../index.php?ed');
        }
    }
}
else {
    // devuelve al menú por error de datos (usuario o clave errados)
    header('Location: ../index.php?ed');
}
}
```

Manual técnico del sistema.

Se verifica que se haya enviado tanto la cédula como la contraseña del usuario y que estos no estén vacíos, en caso de no haberse enviado o estén vacíos, envía al usuario a la pantalla de inicio de sesión, si fueron enviados concatena la nacionalidad con el número de cédula del usuario, y se asigna su valor en la clase Usuario, de igual manera con la contraseña, verifica que exista un usuario con esa cédula y contraseña exactas, e inicia sesión si existe el usuario y se almacenan los datos del usuario en una variable de sesión; en caso de que no, regresa al usuario a la pantalla de inicio de sesión.

```
// si se envia el formulario con las preguntas de seguridad
elseif (isset($_POST['cedula'],$_POST['respuesta_1'],$_POST['respuesta_2'])) {
    // // verifica los datos del usuario
    // $usuarios->set_cedula($_POST['cedula']);
    // if ($datos_usuario = $usuarios->consultar_usuario()) {
    // // si al menos una de las dos preguntas admite el acceso
    // if ((($_POST['respuesta_1'] == $datos_usuario['respuesta_1']) or ($_POST['respuesta_2'] ==
    $datos_usuario['respuesta_2'])) {
    //     session_start();

    //     $_SESSION['datos_login'] = $datos_usuario;
    //     $_SESSION['login'] = true;

    //     $bitacora->set_cedula_usuario($_POST['cedula']);

    //     $_SESSION['id_bitacora'] = $bitacora->iniciar_bitacora();
    //     $_SESSION['acciones'] = "Inicia Sesión (inicio alterno)";

    //     header('Location: ../lobby/index.php');
    // }
    // else {
    //     header('Location: ../index.php?datos_invalidos');
    // }
    // }
    // else {
    //     header('Location: ../index.php?datos_invalidos');
    // }
}
```

También se tiene pensada la implementación de un inicio de sesión a través de preguntas de seguridad. Por los momentos no es funcional.



Manual técnico del sistema.

logout.php

Destruye la sesión del usuario y retorna a la pantalla de inicio de sesión.

```
<?php
    session_start();

    if (!$_SESSION['login']) {
        header('Location: ../index.php');
        exit();
    }

    session_start();

    // destruye la sesion
    session_destroy();

    // redirecciona al menu
    header('Location: ../index.php');

    // finaliza cualquier script php
    exit();
?>
```

Verifica que haya una sesión iniciada. Luego retoma la sesión actual para luego usar la función *session_destroy* para finalizarla, redirige al usuario a la pantalla de inicio de sesión, y por último, finaliza el script.

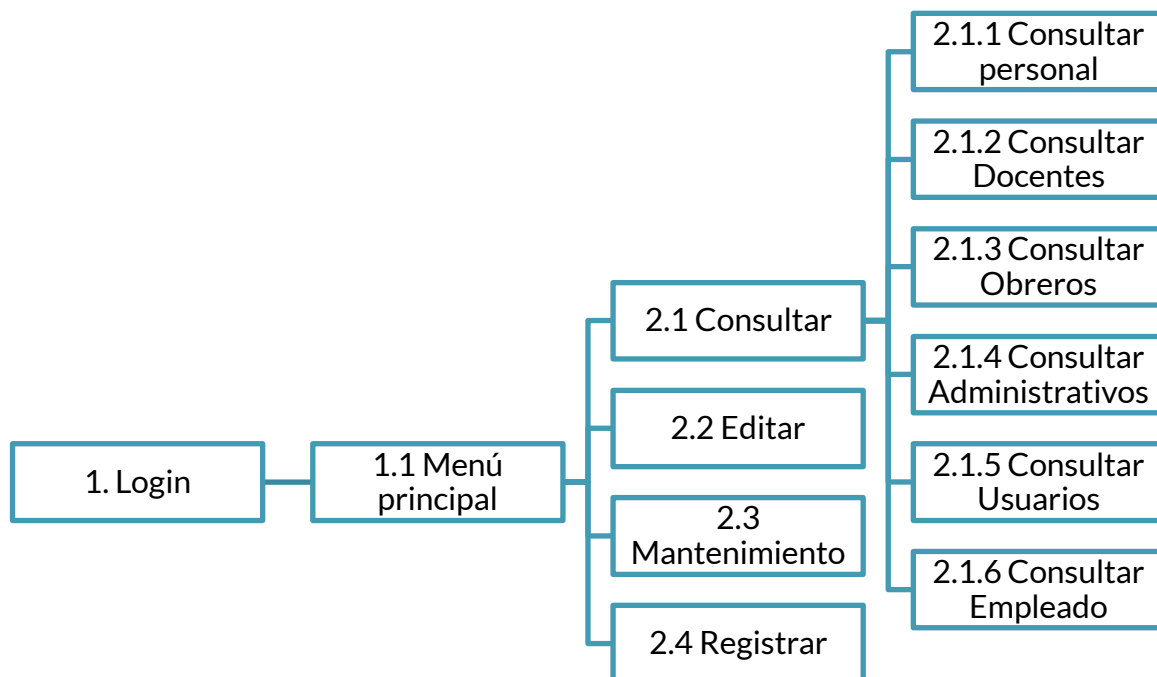


Manual técnico del sistema.

Vistas o pantalla

Las vistas o pantallas de del sistema se encuentran organizadas de la siguiente manera: iniciando con la pantalla de login o inicio de sesión, en la cual se podrá acceder al menú principal si se inicia sesión correctamente. Desde este, se puede ingresar a los cuatro módulos principales del sistema, consultar, editar, y registrar (Por los momentos, la acción del módulo de mantenimiento se encuentra plasmada en el menú principal debido a la poca cantidad de funciones que cumple actualmente).

Diagrama general del sistema.



Manual técnico del sistema.



Login

Se encuentra alojada en index.php dentro de la carpeta sistema.

The screenshot shows a web application interface for login. At the top, there is a dark blue header with a logo on the left and the text 'SISNO-18' in white. The main content area has a light blue background with a pattern of small 'x' marks. In the center, there is a white rounded rectangle with a dark blue border. Inside this rectangle, the title 'Iniciar sesión' is at the top. Below the title, there are three input fields: a dropdown menu labeled 'Nacionalidad' with a downward arrow, a text input field labeled 'Número de cédula', and a text input field labeled 'Contraseña de ingreso'. Below these fields is a dark blue button with the text 'Guardar y continuar' and a right-pointing arrow. At the bottom of the page, there is a dark blue footer with the text 'Hecho con tailwindcss por Elber Rondón'.

En esta primera vista, se muestran tres campos y un botón pertenecientes a un formulario. En este se especifica la nacionalidad (Venezolano / Extranjero), número de cédula y la contraseña del usuario, y se envía con el botón.



Manual técnico del sistema.

```
<body>
  <main class="w-100 h-screen max-h-screen flex flex-col justify-between bg-white fondo-patron overflow-auto">
    <header class="w-100 bg-indigo-dye shadow-lg">
      <nav class="flex flex-row flex-wrap text-white font-semibold">
        <span class="p-3 px-8 bg-cg-blue lg:border-r-4 lg:border-oxford-blue font-extrabold flex justify-center items-center text-lg" title="Sistema de nómina SISNO-18">
          
            SISNO-18
          </span>
        </nav>
      </header>
      <section class="flex flex-row justify-center items-center my-4 lg:my-0">
        <form id="login" action="controladores/login.php" method="post" class="bg-white w-11/12 border-2 border-oxford-blue rounded-3xl max-w-lg">
          <div class="p-2 border-b border-oxford-blue text-center font-bold">
            Iniciar sesión
          </div>
          <div class="max-h-96 overflow-y-auto p-6">
            <div class="menu-inicio p-4">
              <!-- Cédula -->
              <div class="campo col-span-12">
                <label class="form-label" for="cedula">Cédula:</label>
                <div class="grid grid-cols-1 md:grid-cols-3 gap-2">
                  <div class="md:col-span-1">
                    <select id="nacionalidad" class="form-select w-full" name="nacionalidad" required>
                      <option value="" selected disabled>Nacionalidad</option>
                      <option value="V">Venezolano(a)</option>
                      <option value="E">Extranjero(a)</option>
                    </select>
                  </div>
                  <div class="md:col-span-2">
                    <input id="cedula" class="form-input w-full" type="text" name="cedula" minlength="7" maxlength="8" placeholder="Número de cédula" required>
                  </div>
                </div>
              </div>
              <!-- Contraseña -->
              <div class="campo col-span-12">
                <label class="form-label" for="nombre">

```

A nivel de programación, esta vista no presenta mucha complejidad, una estructura html para armar el formulario.

```
</footer>
</main>
<script type="text/javascript" src="js/jquery-3.6.1.min.js"></script>
<script type="text/javascript" src="js/jquery.validate.min.js"></script>
<script type="text/javascript" src="js/messages_es.min.js"></script>
<script type="text/javascript" src="js/login.js"></script>
</body>
</html>
```

En cuanto a librerías. Se llaman las siguientes, se encuentran dentro de la carpeta js del sistema, estas incluyen, jquery y su plugin de validación, el plugin para el lenguaje de los mensajes y las reglas del formulario en login.js.

```
$(document).ready(function() {
  $('#login').submit(function(e) {
    e.preventDefault();
    // or return false;
  });
});

$("#login").validate({
  rules: {
    nacionalidad: {
      required: true,
    },
    cedula: {
      digits: true,
      minlength: 7,
      maxlength: 8,
    },
    contraseña: {
      minlength: 5,
    },
  },
  onfocusout: function(element) {
    this.element(element); // triggers validation
  },
  onkeyup: function(element, event) {
    this.element(element); // triggers validation
  },
  submitHandler: function(form) {
    form.submit();
  },
});
```

Dentro de login.js se comienza declarando una función que espera el evento submit del formulario de inicio de sesión, con el fin de evitar que este se envíe al pulsar enter.

Luego, se inician las validaciones del formulario, especificando las siguientes reglas: la nacionalidad es requerida, la cédula debe contar de entre 7 y 8 dígitos, solo dígitos; la contraseña debe tener una longitud mínima de 5 caracteres.

Luego se especifican los detonantes de la validación del campo al perder el foco o al presionar una tecla y la acción a realizar al momento de enviar.

Todo esto permite que el formulario valide de una manera activa los datos ingresados por el usuario, conforme este llene el formulario recibirá alertas visuales en caso de cometer un error al llenar el campo o dejar vacío un campo obligatorio, como las siguientes:

Cédula:

Nacionalida ▼ sdfsdhf

Este campo es obligatorio. Por favor, escribe sólo dígitos.

Contraseña:

Contraseña de ingreso

Este campo es obligatorio.

Guardar y continuar ➤

Estas mismas desaparecen al cumplir la condición requerida reemplazándola por una tonalidad verde.



Manual técnico del sistema.

Menú principal.

Se encuentra alojado en index.php en la carpeta sistemas/lobby.



Esta vista cuenta con un panel principal con las cartillas gestionar personal, en esta misma muestra el número de empleados registrados y al hacer click sobre esta enviará al usuario al módulo de consulta, la cartilla gestionar usuarios muestra el número de usuarios registrados (solo si quien ingresa es un administrador) y al hacer click en ella envía al usuario al módulo de consulta, con la especificación de consultar usuarios, por último, la cartilla gestionar base de datos, esta cartilla muestra el estado de la base de datos, por los momentos, indica si la base de datos del sistema existe, al hacer click sobre esta va a restaurar los datos de la base de datos a sus valores por defecto con registros de muestra.

Por otro lado, en la parte superior se encuentra la barra de navegación, esa cuenta con las siguientes opciones:

Manual técnico del sistema.

- Inicio: Envía al menú principal
- Personal: Envía al módulo de consulta
- Usuarios: Envía al módulo de consulta con la opción usuarios. (No se muestra usuarios regulares).
- Base de datos: Envía al módulo de mantenimiento. (No se muestra usuarios regulares)
- Cerrar sesión: Inicia el controlador logout.php y cierra la sesión del usuario.

Fuera del maquetado, esta vista cuenta con los siguientes aspectos de programación.

```
<?php
    session_start();

    if (!$SESSION['login']) {
        header('Location: ../index.php');
        exit();
    }

    // var_dump($SESSION);

    require ('../controladores/conexion.php');
    $con = conectar();
    $bd_funciona = comprobar_bd($con);

    require('../clases/empleado.php');

    $emp = new Empleado();

    require('../clases/usuario.php');

    $usu = new Usuario();

?>
```

En un comienzo, se retoma la sesión iniciada en login.php, verifica si el usuario ha iniciado sesión (para evitar ingresar desde la URL sin iniciar sesión). Requiere conexión.php, empleado.php y usuario.php e instancia la conexión a base de datos para luego comprobar el estado de la base de datos, luego objetos de la clase Empleado y Usuario.



Manual técnico del sistema.

```
<div class="max-h-96 overflow-y-auto p-6">
  <div class="border-b mb-4 py-3">
    <p class="text-xl text-center bg-gradient-to-r from-indigo-dye via-cg-blue
to-indigo-dye bg-clip-text">
      Bienvenido al sistema,
      <?php echo $_SESSION['datos_login']['Nombre'].' " '.$_SESSION['datos_login']['
Apellido'];?>.
    </p>
  </div>
```

Muestra el nombre y apellido del usuario en un mensaje de bienvenida.

```
    Gestionar personal.
  </p>
  <p class="p-2 px-4 border-b border-oxford-blue">
    Empleados en nómina:
    <?php echo $emp->contarEmpleados();?>
  </p>
</div>
</a>
```

Muestra el número de empleados registrados.

```
<path stroke="black" stroke-width="1" stroke-dasharray="5 5" />
18.725A7.488 7.488 0 0012 15.75a7.488 7.488 0 00-5.982 2.975m11.963 0a9
9 0 10-11.963 0m11.963 0A8.966 8.966 0 0112 21a8.966 8.966 0
01-5.982-2.275M15 9.75a3 3 0 11-6 0 3 3 0 016 0z" />
</svg>
    Gestionar usuarios.
  </p>
  <p class="p-2 px-4 border-b border-oxford-blue">
    Usuarios registrados:
    <?php echo $usu->contarUsuarios(); ?>
  </p>
</div>
</a>
<a class="m-3" href="/controladores/base-datos.php">
```

Manual técnico del sistema.

Muestra el número de usuarios registrados.

```
</p>
<p class="p-2 px-4 border-b border-oxford-blue font-semibold text-sm">
  Estatus de BD:
  <span class="p-2 font-normal ml-2">
    <?php if ($bd_funciona): ?>
      Funcionando
      <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
        stroke-width="1.5" stroke="green" class="w-5 inline-block">
        <path stroke-linecap="round" stroke-linejoin="round" d="M9 12.75L11.25
          15 15 9.75M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />
      </svg>

    <?php else: ?>
      No funcionando.
      <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
        stroke-width="1.5" stroke="red" class="w-6 inline-block">
        <path stroke-linecap="round" stroke-linejoin="round" d="M9.75 9.75L4.5
          4.5M0-4.5L-4.5 4.5M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />
      </svg>

    <?php endif ?>
  </span>
</p>
```

Muestra con un condicional el estado de la base de datos (Funcionando / No funcionando).



Manual técnico del sistema.

Consultar.

A primera vista, este módulo muestra el personal registrado en el sistema, así como los usuarios.

Este módulo se encuentra en la carpeta consulta y cuenta con cinco variantes y un submódulo que permite hacer una consulta individual y específica del empleado.

Nota: todas las secciones a excepción de consultar usuarios cuentan con la opción de acceder al módulo de registrar empleados y consultar empleado. A través de los botones consultar y registrar empleado

A nivel de programación este módulo hace las siguientes acciones:

```
<?php
session_start();

if (!$_SESSION['login']) {
    header('Location: ../../index.php');
    exit();
}

if (isset($_GET['con'])) {
    if ($_GET['con'] == "obr") {
        $con = 0;
        $title = "obreros";
    }
    elseif ($_GET['con'] == "doc") {
        $con = 1;
        $title = "docentes";
    }
    elseif ($_GET['con'] == "adm") {
        $con = 2;
        $title = "personal administrativo";
    }
    elseif ($_GET['con'] == "usu") {
        $con = 3;
        $title = "usuarios";
    }
    elseif ($_GET['con'] == "per") {
        $con = 4;
        $title = "personal";
    }
    elseif ($_GET['con'] == "emp" and isset($_POST['id_Persona'], $_POST['tipo_empleado'])) {
        $con = 5;
        $title = "empleado";
    }
    else {
        $con = 4;
        $title = "personal";
    }
}
else {
    $con = 4;
    $title = "personal";
}
?>
```

Manual técnico del sistema.

Inicia validando la sesión del usuario, luego valida la variable pasada mediante GET para seleccionar que tipo de datos mostrar, en el siguiente orden, obreros, docentes, administrativos, usuarios, personal, consulta de empleado, y en los casos de que la variable get sea una fuera de las opciones el valor será consulta de personal, de igual manera si no fue enviada una variable GET.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../../css/estilos.css">
  <link rel="shortcut icon" href="../../img/sisno-18.png" type="image/png">
  <title>Consultar <?php echo $title; ?></title>
</head>
<body>
```

La variable \$title será usada para indicar en la pestaña del navegador en que sección de consulta se encuentra el usuario.

```
</a>
<a class="item-navbar <?php if($con!=3) {echo "active";}?" href="index.php">
  <svg class="w-6 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor">
    <path stroke-linecap="round" stroke-linejoin="round" d="M20.25 14.15v4.25c0 1.094-.787 2.036-1.872 2.18-2.087 2.77-4.216 4.26-6.378 4.25-4.291-.143-6.378-.42c-1.085-.144-1.872-1.086-1.872-2.18v-4.25m16.5 0a2.18 2.18 0 00-.75-1.661V8.706c0-1.081-.768-2.015-1.837-2.175a48.114 48.114 0 00-3.413-.387m4.5 8.006c-.194 1.65-.42 2.95-.673 3.8A23.978 23.978 0 0112 15.75c-2.648 0-5.195-.429-7.577-1.22a2.016 2.016 0 01-.673-.38m0 0A2.18 2.18 0 0112 12.489V8.706c0-1.081-.768-2.015-1.837-2.175a48.111 48.111 0 013.413-.387m7.5 0V5.25A2.25 2.25 0 0013.5 3h-3a2.25 2.25 0 00-2.25 2.25v.894m7.5 0a48.667 48.667 0 00-7.5 0M12 12.75h.008v.008H12v-.008z" />
  </svg>
  <span>Personal</span>
</a>
<?php if ($_SESSION['datos_login']['Rol'] == 'Administrador'): ?>
  <a class="item-navbar <?php if($con==3) {echo "active";}?" href="index.php?con=usu">
    <svg class="w-6 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor">
      <path stroke-linecap="round" stroke-linejoin="round" d="M17.982 18.725A7.488 7.488 0 0012 15.75a7.488 7.488 0 00-5.982 2.975m11.963 0a9 9 0 0111.963 0A8.966 8.966 0 0112 21a8.966 8.966 0 01-5.982-2.275M15 9.75a3 3 0 11-6 0 3 3 0 016 0z" />
    </svg>
    <span>Usuarios</span>
  </a>
  <a class="item-navbar" href="../../mantenimiento/index.php">
    <svg class="w-6 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor">
      <path stroke-linecap="round" stroke-linejoin="round" d="M20.25 6.375c0 2.278-3.694 4.125-8.25 4.125S3.75 8.653 3.75 6.375m16.5 0c0-2.278-3.694-4.125-8.25-4.125S3.75 4.097 3.75 6.375m16.5 0v11.25c0 2.278-3.694 4.125-8.25 4.125s-8.25-1.847-8.25-4.125V6.375m16.5 0v3.75c0 3.75 3.75 3.75 3.75 3.75c0 2.25 16.153 16.556 18 12 18-8.25-1.847-8.25-4.125V6.375m16.5 0c0 2.278-3.694 4.125-8.25 4.125s-8.25-1.847-8.25-4.125" />
    </svg>
    <span>Base de datos</span>
  </a>
<?php endif; ?>
```



Manual técnico del sistema.

La variable \$con será usada para especificar la consulta activa en la barra de navegación, resaltando la pestaña personal siempre y cuando no se esté consultando usuarios, en ese caso se resaltarán la pestaña usuarios.

```
</div>
<?php
    if ($con == 0){
        require 'obreros.php';
    }
    elseif($con == 1){
        require 'docentes.php';
    }
    elseif($con == 2){
        require 'administrativos.php';
    }
    elseif($con == 3){
        require 'usuarios.php';
    }
    elseif($con == 4){
        require 'personal.php';
    }
    elseif($con == 5){
        require 'consultar-empleado.php';
    }
}
?>
</div>
```

Asimismo, se usa esta variable para decidir que variante del módulo se va a usar.

Manual técnico del sistema.

Consulta de personal

En el caso de que \$con sea igual a 4, se requiere el archivo personal.php como un agregado a index.php



The screenshot shows the SISNO-18 web application interface. The top navigation bar includes the SISNO-18 logo, a home icon, a 'Personal' menu item, and icons for user profile, database, and power. The main content area is titled 'Área de consulta' and contains a 'Selector de consulta' with buttons for 'Consultar Personal', 'Consultar Obreros', 'Consultar Docentes', 'Consultar Administrativos', and 'Consultar Usuarios'. Below this is the 'Consulta de personal.' section, which includes a 'Registrar empleado' button and a table of employee data.

Nombre	Apellido	Cédula	Fecha de nacimiento	Tiempo en nómina	Tipo de personal	Acciones
Morganica	Lawty	V12649473	2022-04-14	5 Meses	Obrero	consultar 🔍
Edythe	Rigney	V13415496	2022-01-16	5 Meses	Obrero	consultar 🔍
Berne	Pavlenkov	V84769533	2022-08-09	5 Meses	Obrero	consultar 🔍

Below the table is the 'Sistema de nómina' section. The footer of the application states 'Hecho con tailwindcss por Elber Rondón'.

Muestra los datos de generales de los empleados registrados en sistema, teniendo en cuenta nombres, apellidos, cédula, fecha de nacimiento, cuánto tiempo lleva en nómina de una manera aproximada y que tipo de personal es.

A nivel de programación, hace lo siguiente:

```
<?php
require('../controladores/conexion.php');
require('../clases/empledo.php');
require('../clases/calculos.php');

$emp = new Empleado();
$calc = new Calculo();

$lista_personal = $emp->mostrar();

?>
```

Manual técnico del sistema.

Inicia requiriendo el controlador de conexión a base de datos y los archivos de empleado y cálculos, instancia estas clases y usa el método mostrar de la clase Empleado para almacenar la lista de empleados registrados.

```
<p class="text-xl text-center font-semibold mt-5 mb-3">Consulta de personal.</p>
<a class="boton mb-2" href="../../registrar/paso-1.php">
  Registrar empleado
  <svg class="w-6 ml-1 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor">
    <path stroke-linecap="round" stroke-linejoin="round" d="M19 7.5v3m0 0v3m0-3h3m-3 0h-3m-2.25-4.125a3.375 3.375 0 11-6.75 0 3.375 3.375 0 016.75 0zM4 19.235v-.11a6.375 6.375 0 0112.75 0v.109A12.318 12.318 0 0110.374 21c-2.331 0-4.512-.645-6.374-1.766z" />
  </svg>
</a>
```

The image shows a Sublime Text editor window with the following content:

```
C:\xampp\htdocs\sistema_ing_soft_II\sistema\lobby\consultar\personal.php (sistema_ing_soft_II) - Sublime Text (UNREGISTERED)
index.php | personal.php x
<table id="personal" class="table-auto w-full border-separate border border-indigo-dye">
  <thead>
    <tr>
      <th class="border bg-indigo-dye text-white p-1">Nombre</th>
      <th class="border bg-indigo-dye text-white p-1">Apellido</th>
      <th class="border bg-indigo-dye text-white p-1">Cédula</th>
      <th class="border bg-indigo-dye text-white p-1">Fecha de nacimiento</th>
      <th class="border bg-indigo-dye text-white p-1">Tiempo en nomina</th>
      <th class="border bg-indigo-dye text-white p-1">Tipo de personal</th>
      <th class="border bg-indigo-dye text-white p-1">Acciones</th>
    </tr>
  </thead>
  <tbody class="text-center">
    <tr>
      <td colspan="7">
        <?php foreach ($lista_personal as $empleado): ?>
          <tr>
            <td class="border border-indigo-dye p-1"><?php echo $empleado['Nombre']; ?></td>
            <td class="border border-indigo-dye p-1"><?php echo $empleado['Apellido']; ?></td>
            <td class="border border-indigo-dye p-1"><?php echo $empleado['Cedula']; ?></td>
            <td class="border border-indigo-dye p-1"><?php echo $empleado['Fecha_Nac']; ?></td>
            <td class="border border-indigo-dye p-1"><?php echo $empleado['Fecha_Ingreso']; ?></td>
            <td colspan="2">
              <?php
                $emp->setid($empleado['id_empleado']);
                $tipo_empleado = $emp->verificarTipo();

                if ($tipo_empleado == 0) {
                  $st = "Obrero";
                }
                elseif ($tipo_empleado == 1) {
                  $st = "Docente";
                }
                elseif ($tipo_empleado == 2) {
                  $st = "Administrativo";
                }
              <?>
            <td class="border border-indigo-dye p-1"><?php echo $st; ?></td>
          </tr>
          <tr>
            <td colspan="7">
              <form action="index.php?con=emp" method="post">
                <input type="hidden" name="id_Persona" value="<?php echo $empleado['id_Persona']; ?>">
                <input type="hidden" name="tipo_empleado" value="<?php echo $tipo_empleado; ?>">
                <button class="boton p-1 px-2 text-sm" type="submit">
                  consultar
                <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" class="w-5 h-5">
                  <path stroke-linecap="round" stroke-linejoin="round" d="M21 21 5.197 5.197M8 7.5 15.197 15.197M8 12.5 15.197 12.5" />
                </svg>
              </button>
            </td>
          </tr>
        </tr>
      </td>
    </tr>
  </tbody>
</table>
```

The status bar at the bottom indicates: 04:34:04, 3K, Line 12, Column 3. The right sidebar shows a file explorer with a tree view of the project structure.

Luego de eso, se imprimirá en pantalla una barra de título con un botón que permite registrar un empleado, seguida por la tabla de empleados, mostrando entre los resultados obtenidos los



Manual técnico del sistema.

datos: nombre, apellido, cédula, fecha de nacimiento, tiempo en nómina tipo de personal y una columna de acciones.

En el caso del tiempo en nómina, es usado el método `diferenciaF()` para calcular el tiempo aproximado que lleva en nómina el empleado.

Para el tipo de empleado, se usa el método `verificarTipo` y se evalúa con un `if`, para mostrar si el empleado es obrero, docente o administrativo; almacenándolo en la variable `$t`.

En la columna de acción se encuentra un formulario con un botón que envía el identificador y el tipo de empleado al mismo `index.php` pasando una variable GET que especifica que se va a consultar ese empleado.



Manual técnico del sistema.

Consulta de obreros

En el caso de que \$con sea igual a 0, se requiere el archivo obreros.php como un agregado a index.php

The screenshot displays the SISNO-18 web application interface. At the top, there is a dark blue header with the SISNO-18 logo, a home icon, a 'Personal' menu item, and icons for user, database, and power. Below the header, the main content area is titled 'Área de consulta'. It features a 'Selector de consulta' with five buttons: 'Consultar Personal', 'Consultar Obreros' (which is highlighted), 'Consultar Docentes', 'Consultar Administrativos', and 'Consultar Usuarios'. Below this, the section is titled 'Consulta de obreros.' and includes a 'Registrar empleado' button with a plus icon. A table lists employee data with columns: Nombre, Apellido, Cédula, Fecha de nacimiento, Rol de obrero, Tiempo en nomina, and Acciones. The table contains three rows of data. Each row has a 'consultar' button with a magnifying glass icon. At the bottom of the table area, it says 'Sistema de nómina'. The footer of the page states 'Hecho con tailwindcss por Elber Rondón'.

Nombre	Apellido	Cédula	Fecha de nacimiento	Rol de obrero	Tiempo en nomina	Acciones
Morganica	Lawty	V12649473	2022-04-14	Limpieza	5 Meses	consultar 🔍
Edythe	Rigney	V13415496	2022-01-16	Cocina	5 Meses	consultar 🔍
Berne	Pavlenkov	V84769533	2022-08-09	Limpieza	5 Meses	consultar 🔍

Muestra los datos de generales de los empleados obreros registrados en sistema, teniendo en cuenta nombres, apellidos, cédula, fecha de nacimiento, el rol o área que ocupa como obrero y cuánto tiempo lleva en nómina de una manera aproximada.

Manual técnico del sistema.

A nivel de programación, hace lo siguiente:

```
<?php
    require('../..../controladores/conexion.php');
    require('../..../clases/calculos.php');
    require('../..../clases/obrero.php');

    $ob = new Obrero();
    $calc = new Calculo();

    $lista_ob = $ob->mostrar();

?>

<p class="text-xl text-center font-semibold mt-5 mb-3">Consulta de obreros.</p>

<a class="boton mb-2" href="../registrar/paso-1.php">
    Registrar empleado
    <svg class="w-6 ml-1 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="
    currentcolor">
        <path stroke-linecap="round" stroke-linejoin="round" d="M19 7.5v3m0 0v3m-3 0h-3m-2.25-4.125a3.375 3.375 0 11-6.75 0 3.375
        3.375 0 016.75 0z" />
    </svg>
</a>
```

```
<table id="obrerros" class="table-auto w-full border-separate border border-indigo-dye">
    <thead>
        <tr>
            <th class="border bg-indigo-dye text-white p-1">Nombre</th>
            <th class="border bg-indigo-dye text-white p-1">Apellido</th>
            <th class="border bg-indigo-dye text-white p-1">Cédula</th>
            <th class="border bg-indigo-dye text-white p-1">Fecha de nacimiento</th>
            <th class="border bg-indigo-dye text-white p-1">Rol de obrero</th>
            <th class="border bg-indigo-dye text-white p-1">Tiempo en nomina</th>
            <th class="border bg-indigo-dye text-white p-1">Acciones</th>
        </tr>
    </thead>
    <tbody class="text-center">
        <?php foreach ($lista_ob as $Obrero): ?>
            <tr>
                <td class="border border-indigo-dye p-1"><?php echo $Obrero['Nombre']; ?></td>
                <td class="border border-indigo-dye p-1"><?php echo $Obrero['Apellido']; ?></td>
                <td class="border border-indigo-dye p-1"><?php echo $Obrero['Cedula']; ?></td>
                <td class="border border-indigo-dye p-1"><?php echo $Obrero['Fecha_Nac']; ?></td>
                <td class="border border-indigo-dye p-1"><?php echo $Obrero['Rol']; ?></td>
                <td class="border border-indigo-dye p-1"><?php echo $calc->diferenciaF($Obrero['Fecha_Ingreso']); ?></td>
                <td class="border border-indigo-dye p-1">
                    <form action="index.php?con=emp" method="post">
                        <input type="hidden" name="id_Persona" value="<?php echo $Obrero['id_Persona']; ?>">
                        <input type="hidden" name="tipo_empleado" value="0">
                        <button class="boton p-1 px-2 text-sm" type="submit">
                            consultar
                            <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="
                            currentColor" class="w-5 ml-1 inline-block">
                                <path stroke-linecap="round" stroke-linejoin="round" d="M21 21l-5.197-5.197m0 0A7.5 7.5 0
                                105.196 5.196a7.5 7.5 0 0010.607 10.607z" />
                            </svg>
                        </button>
                    </form>
                </td>
            </tr>
        <?php endforeach ?>
    </tbody>
</table>
```

Inicia requiriendo el controlador de conexión a base de datos y los archivos de obrero y cálculos, instancia estas clases y usa el método mostrar de la clase Obrero para almacenar la lista de obreros registrados.



Manual técnico del sistema.

Luego de eso, se imprimirá en pantalla una barra de título con un botón que permite registrar un empleado, seguida por la tabla de obreros, mostrando entre los resultados obtenidos los datos: nombre, apellido, cédula, fecha de nacimiento el rol de obrero, el tiempo en nómina y una columna de acciones.

En el caso del tiempo en nómina, es usado el método `diferenciaF()` para calcular el tiempo aproximado que lleva en nómina el empleado.

En la columna de acción se encuentra un formulario con un botón que envía el identificador y el tipo de empleado al mismo `index.php` pasando una variable GET que especifica que se va a consultar ese empleado.



Manual técnico del sistema.

Consulta de docentes

En el caso de que \$con sea igual a 1, se requiere el archivo docentes.php como un agregado a index.php

The screenshot displays the SISNO-18 web application interface. At the top, there is a dark blue header with the SISNO-18 logo and navigation icons for Home, Personal, and other functions. Below the header, the main content area is titled 'Área de consulta'. Under this title, there is a 'Selector de consulta' section with five buttons: 'Consultar Personal', 'Consultar Obreros', 'Consultar Docentes' (which is highlighted), 'Consultar Administrativos', and 'Consultar Usuarios'. Below the selector, there is a 'Consulta de docentes.' section with three buttons: 'Registrar empleado' (with a plus icon), 'Descargar Excel' (with a download icon), and 'Descargar PDF' (with a download icon). The main part of the interface is a table with the following data:

Nombre	Apellido	Cédula	Fecha de nacimiento	Horas académicas	Tiempo en nomina	Área	Acciones
Terri	Fist	V93909125	2022-10-18	16 Horas semanales	5 Meses	Inglés	consultar 🔍
Ramsay	Roussel	V69823294	2022-07-21	26 Horas semanales	5 Meses	Castellano	consultar 🔍
Frannie	Pirkis	V10113385	2022-08-06	18 Horas semanales	5 Meses	Matemática	consultar 🔍

At the bottom of the interface, there is a footer that reads 'Hecho con tailwindcss por Elber Rondón'.

Muestra los datos de generales de los docentes registrados en sistema, teniendo en cuenta nombres, apellidos, cédula, fecha de nacimiento, las horas académicas semanales, cuánto tiempo lleva en nómina de una manera aproximada y que área imparte.

A nivel de programación, hace lo siguiente:

```

<?php

require('.././controladores/conexion.php');
require('.././clases/docente.php');
require('.././clases/calculos.php');

$doc = new Docente();
$scal = new Calculo();

$lista_doc = $doc->mostrar();

?>

<p class="text-xl text-center font-semibold mt-5 mb-3">Consulta de docentes.</p>

<a class="boton mb-2" href="../registrar/paso-1.php">
  Registrar empleado
  <svg class="w-6 ml-1 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5"
  stroke="currentColor">
    <path stroke-linecap="round" stroke-linejoin="round" d="M19 7.5v3m0 0v3m0-3h3m-3 0h-3m-2.25-4.125a3.375 3.375 0 11-6.75 0
    3.375 3.375 0 016.75 0z" />
  </svg>
</a>

<a class="boton mb-2" href="../controladores/docente-excel.php">
  Descargar Excel
  <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" class="w-6
  ml-2 inline-block">
    <path stroke-linecap="round" stroke-linejoin="round" d="M3 16.5v2.25A2.25 2.25 0 005.25 21h13.5A2.25 2.25 0 0021
    18.75V16.5" />
  </svg>
</a>

<a class="boton mb-2" href="../controladores/docente-pdf.php" target="_blank">
  Descargar PDF
  <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" class="w-6
  ml-2 inline-block">
    <path stroke-linecap="round" stroke-linejoin="round" d="M3 16.5v2.25A2.25 2.25 0 005.25 21h13.5A2.25 2.25 0 0021
    18.75V16.5" />
  </svg>
</a>

```

```

<table id="docentes" class="table-auto w-full border-separate border border-indigo-dye text-sm">
  <thead>
    <tr>
      <th class="border bg-indigo-dye text-white p-1">Nombre</th>
      <th class="border bg-indigo-dye text-white p-1">Apellido</th>
      <th class="border bg-indigo-dye text-white p-1">Cédula</th>
      <th class="border bg-indigo-dye text-white p-1">Fecha de nacimiento</th>
      <th class="border bg-indigo-dye text-white p-1">Horas académicas</th>
      <th class="border bg-indigo-dye text-white p-1">Tiempo en nomina</th>
      <th class="border bg-indigo-dye text-white p-1">Área</th>
      <th class="border bg-indigo-dye text-white p-1">Acciones</th>
    </tr>
  </thead>
  <tbody class="text-center">
    <?php foreach ($lista_doc as $docente): ?>
      <tr>
        <td class="border border-indigo-dye p-1"><?php echo $docente['Nombre']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $docente['Apellido']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $docente['Cedula']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $docente['Fecha_Nac']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $docente['Horas_Clase_S']; ?> Horas semanales</td>
        <td class="border border-indigo-dye p-1"><?php echo $scal->diferenciaF($docente['Fecha_Ingreso']); ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $docente['Area']; ?></td>
        <td class="border border-indigo-dye p-1">
          <form action="index.php?con=emp" method="post">
            <input type="hidden" name="id_Persona" value="<?php echo $docente['id_Persona']; ?>">
            <input type="hidden" name="tipo_empleado" value="1">
            <button class="boton p-1 px-2 text-sm" type="submit">
              consultar
              <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor"
              class="w-5 ml-1 inline-block">
                <path stroke-linecap="round" stroke-linejoin="round" d="M21 21l-5.197-5.197m0 0a7.5 7.5 0 10 10.607 0
                7.5 7.5 0 00-10.607 0" />
              </svg>
            </button>
          </form>
        </td>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>

```



Manual técnico del sistema.

Inicia requiriendo el controlador de conexión a base de datos y los archivos de docente y cálculos, instancia estas clases y usa el método mostrar de la clase Docente para almacenar la lista de docentes registrados.

Luego de eso, se imprimirá en pantalla una barra de título con un botón que permite registrar un empleado y botones para generar los reportes de los docentes en formato Excel y Pdf, seguida por la tabla de docentes, mostrando entre los resultados obtenidos los datos: nombres, apellidos, cédula, fecha de nacimiento, las horas académicas semanales, cuánto tiempo lleva en nómina de una manera aproximada, el área que imparte y una columna de acciones.

En el caso del tiempo en nómina, es usado el método diferenciaF() para calcular el tiempo aproximado que lleva en nómina el empleado.

En la columna de acción se encuentra un formulario con un botón que envía el identificador y el tipo de empleado al mismo index.php pasando una variable GET que especifica que se va a consultar ese empleado.



Manual técnico del sistema.

Consulta de personal administrativo

The screenshot displays the SISNO-18 web application interface. At the top, there is a dark blue header with the SISNO-18 logo, a home icon, a 'Personal' tab, and icons for user, database, and power. Below the header, the main content area is titled 'Área de consulta'. Under this, there is a 'Selector de consulta' section with five buttons: 'Consultar Personal', 'Consultar Obreros', 'Consultar Docentes', 'Consultar Administrativos', and 'Consultar Usuarios'. The 'Consultar Administrativos' button is selected. Below the selector, the text 'Consulta de personal administrativo.' is displayed. To the left of a table is a button 'Registrar empleado' with a plus icon. The table has six columns: 'Nombre', 'Apellido', 'Cédula', 'Fecha de nacimiento', 'Tiempo en nomina', and 'Acciones'. It contains three rows of data. Each row has a 'consultar' button with a magnifying glass icon in the 'Acciones' column. At the bottom of the table area, the text 'Sistema de nómina' is visible. The footer of the application is a dark blue bar with the text 'Hecho con tailwindcss por Elber Rondón'.

Nombre	Apellido	Cédula	Fecha de nacimiento	Tiempo en nomina	Acciones
Cal	Vedyashkin	V66298340	2021-12-07	5 Meses	consultar 🔍
Ralf	Barnwall	V50950642	2022-10-07	5 Meses	consultar 🔍
Rebekkah	Jaycock	V12716584	2022-10-18	2 Años	consultar 🔍

Muestra los datos de generales del personal administrativo registrado en sistema, teniendo en cuenta nombres, apellidos, cédula, fecha de nacimiento y cuánto tiempo lleva en nómina de una manera aproximada.

A nivel de programación, hace lo siguiente:

Manual técnico del sistema.

```
<?php
require('../..../controladores/conexion.php');
require('../..../clases/administrativo.php');
require('../..../clases/calculos.php');

$calc = new Calculo();

$adm = new Administrativo();

$lista_adm = $adm->mostrar();
?>

<p class="text-xl text-center font-semibold mt-5 mb-3">Consulta de personal administrativo.</p>

<a class="boton mb-2" href="../../registrar/paso-1.php">
  Registrar empleado
  <svg class="w-6 ml-1 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
    stroke-width="1.5" stroke="currentColor">
    <path stroke-linecap="round" stroke-linejoin="round" d="M19 7.5v3m0 0v3m-3 0h-3m-2.25-4.125a3.375
      3.375 0 11-6.75 0 3.375 3.375 0 016.75 0zM4 19.235v-.11a6.375 6.375 0 0112.75 0v.109A12.318 12.318 0
      0110.374 21c-2.331 0-4.512-.645-6.374-1.766z" />
    </svg>
  </a>
```

```
<table id="administrativos" class="table-auto w-full border-separate border border-indigo-dye">
  <thead>
    <tr>
      <th class="border bg-indigo-dye text-white p-1">Nombre</th>
      <th class="border bg-indigo-dye text-white p-1">Apellido</th>
      <th class="border bg-indigo-dye text-white p-1">Cédula</th>
      <th class="border bg-indigo-dye text-white p-1">Fecha de nacimiento</th>
      <th class="border bg-indigo-dye text-white p-1">Tiempo en nomina</th>
      <th class="border bg-indigo-dye text-white p-1">Acciones</th>
    </tr>
  </thead>
  <tbody class="text-center">
    <?php foreach ($lista_adm as $Administrativo): ?>
      <tr>
        <td class="border border-indigo-dye p-1"><?php echo $Administrativo['Nombre']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $Administrativo['Apellido']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $Administrativo['Cedula']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $Administrativo['Fecha_Nac']; ?></td>
        <td class="border border-indigo-dye p-1"><?php echo $calc->diferenciaF($Administrativo['Fecha_Ingreso']);
          ?></td>
        <td class="border border-indigo-dye p-1">
          <form action="index.php?con=emp" method="post">
            <input type="hidden" name="id_Persona" value="<?php echo $Administrativo['id_Persona']; ?>">
            <input type="hidden" name="tipo_empleado" value="2">
            <button class="boton p-1 px-2 text-sm" type="submit">
              consultar
              <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="
                currentColor" class="w-5 ml-1 inline-block">
                <path stroke-linecap="round" stroke-linejoin="round" d="M21 21l-5.197-5.197m0 0A7.5 7.5 0
                  105.196 5.196a7.5 7.5 0 0010.607 10.607z" />
              </svg>
            </button>
          </form>
        </td>
      </tr>
    <?php endforeach ?>
  </tbody>
</table>
```

Inicia requiriendo el controlador de conexión a base de datos y los archivos de administrativo y cálculos, instancia estas clases y usa el método mostrar de la clase Administrativo para almacenar la lista del personal administrativo registrado.



Manual técnico del sistema.

Luego de eso, se imprimirá en pantalla una barra de título con un botón que permite registrar un empleado y luego mostrando entre los resultados obtenidos los datos: nombres, apellidos, cédula, fecha de nacimiento, cuánto tiempo lleva en nómina de una manera aproximada y una columna de acciones.

En el caso del tiempo en nómina, es usado el método `diferenciaF()` para calcular el tiempo aproximado que lleva en nómina el empleado.

En la columna de acción se encuentra un formulario con un botón que envía el identificador y el tipo de empleado al mismo `index.php` pasando una variable GET que especifica que se va a consultar ese empleado.



Manual técnico del sistema.

Consulta de usuarios

 **SISNO-18**

 Usuarios

Área de consulta

Selector de consulta

Consultar Personal

Consultar Obreros

Consultar Docentes

Consultar Administrativos

Consultar Usuarios

Consulta de usuarios.

Nombre	Apellido	Cédula	Fecha de nacimiento	Rol
Elber	Rondón	V27919566	2001-05-05	Administrador
user	user	V11111111	0000-00-00	Usuario

Sistema de nómina

Hecho con  tailwindcss por Elber Rondón

Muestra los datos de generales de los usuarios registrados en sistema, teniendo en cuenta nombres, apellidos, cédula, fecha de nacimiento y su rol de usuario.

Manual técnico del sistema.

A nivel de programación, hace lo siguiente:

```
<?php
require('.././controladores/conexion.php');
require('.././clases/persona.php');
require('.././clases/usuario.php');
require('.././clases/calculos.php');

$usu = new Usuario();
$calc = new Calculo();

$lista_usuario = $usu->mostrar();
?>

<?php if ($_SESSION['datos_login']['Rol'] == 'Administrador'): ?>
<p class="text-xl text-center font-semibold mt-5 mb-3">Consulta de usuarios.</p>

<!-- <a class="boton mb-2" href="..registrar/paso-1.php">
Registrar empleado
<svg class="w-6 ml-1 inline-block" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5"
stroke="currentColor">
<path stroke-linecap="round" stroke-linejoin="round" d="M19 7.5v3m0 0v3m-3 0h-3m-2.25-4.125a3.375 3.375 0 11-6.75
0 3.375 3.375 0 016.75 0zM4 19.25v-.11a6.375 6.375 0 112.75 0v.109A12.318 12.318 0 0110.374 21c-2.331
0-4.512-.645-6.374-1.766z"/>
</svg>
</a> -->

<table id="personal" class="table-auto w-full border-separate border border-indigo-dye">
<thead>
<tr>
<th class="border bg-indigo-dye text-white p-1">Nombre</th>
<th class="border bg-indigo-dye text-white p-1">Apellido</th>
<th class="border bg-indigo-dye text-white p-1">Cédula</th>
<th class="border bg-indigo-dye text-white p-1">Fecha de nacimiento</th>
<th class="border bg-indigo-dye text-white p-1">Rol</th>
</tr>
</thead>
<tbody class="text-center">
<?php foreach ($lista_usuario as $usuario): ?>
<tr>
<td class="border border-indigo-dye p-1"><?php echo $usuario['Nombre']; ?></td>
<td class="border border-indigo-dye p-1"><?php echo $usuario['Apellido']; ?></td>
<td class="border border-indigo-dye p-1"><?php echo $usuario['Cedula']; ?></td>
<td class="border border-indigo-dye p-1"><?php echo $usuario['Fecha_Nac']; ?></td>
<td class="border border-indigo-dye p-1"><?php echo $usuario['Rol']; ?></td>
</tr>
<?php endforeach ?>
</tbody>
</table>
<?php endif ?>
```

Inicia requiriendo el controlador de conexión a base de datos y los archivos de usuario y cálculos, instancia estas clases y usa el método mostrar de la clase Usuario para almacenar la lista de usuarios registrados.

Luego de eso, se imprimirá en pantalla una barra de título con un botón que permite registrar un usuario (No funcional por los momentos) y luego mostrando entre los resultados obtenidos los datos: nombres, apellidos, cédula, fecha de nacimiento, el rol de usuario y una columna de acciones.



Manual técnico del sistema.

Cabe mencionar que la tabla solo se mostrará si el usuario que solicita la consulta es un administrador. De lo contrario, no lo hará.

Manual técnico del sistema.



Consultar empleado



Muestra todos los datos pertenecientes a un empleado. (Véase clases del sistema)

A nivel de programación hace lo siguiente:

```
consultar-empleado.php

<?php

require('../controladores/conexion.php');
require('../clases/calculos.php');

require('../clases/contacto.php');
require('../clases/telefono.php');
require('../clases/direccion.php');
require('../clases/informe.php');
require('../clases/estudio.php');
require('../clases/carga-horaria.php');

if ($_POST['tipo_empleado'] == 0) {
    include('../clases/obrero.php');
    $emp = new Obrero();
    $empleado = $emp->consultar($_POST['id_Persona']);
}
elseif ($_POST['tipo_empleado'] == 1) {
    include('../clases/docente.php');
    $emp = new Docente();
    $empleado = $emp->consultar($_POST['id_Persona']);
}
elseif ($_POST['tipo_empleado'] == 2) {
    include('../clases/administrativo.php');
    $emp = new Administrativo();
    $empleado = $emp->consultar($_POST['id_Persona']);
}

$con = new Conexion();
$conectar = $con->consultar($_POST['id_Persona']);

$tel = new Telefono();
$telefonos = $tel->consultar($conectar['id_Contacto']);

$dir = new Direccion();
$direccion = $dir->consultar($conectar['id_Contacto']);

$inf = new Informe();
$informe = $inf->consultar($_POST['id_Persona']);

$est = new Estudio();
$estudio = $est->consultar($empleado['id_Empleado']);

$ch = new Carga_Horaria();
$carga = $ch->consultar($empleado['id_Empleado']);

$calc = new Calculo();

?>
```

Inicia requiriendo el controlador de conexión a base de datos, luego los archivos de cálculos, contacto, teléfono, dirección, informe, estudio, y carga horaria para instanciar sus clases correspondientes.

Según el tipo de empleado especificado, instanciará la clase Obrero, Docente o Administrativo, así como también consultará los datos del empleado asociados a esa clase en la base de datos.

Hará consultas de igual manera con el resto de clases hasta traer todos los datos correspondientes al empleado y finalmente instancia la clase Calculo.



Manual técnico del sistema.

```
C:\xampp\htdocs\sistema_ing_soft_II\sistema\lobby\consultar\consultar-empleado.php • (sistema_ing_soft_II) - Sublime Text (UNRE...  
consultar-empleado.php x  
51 <div class="w-2/3 border-2 border-indigo-dye mx-auto mt-5 rounded-2xl p-4">  
52 <div class="grid grid-cols-1">  
53 <div class="p-1">  
54 <p class="text-center">  
55 Información de  
56 <?php echo Sempleado['Nombre']. " ". Sempleado['Apellido']; ?>  
57 </p>  
58 </div>  
59 <div class="p-4 col-span-1 flex justify-center items-center">  
60 "  
64 </div>  
65 </div>  
66 <div class="p-2 px-5">  
67 <p>  
68 <span class="font-semibold">  
69 Nombre completo:  
70 </span>  
71 <?php echo Sempleado['Nombre']. " ". Sempleado['Apellido']; ?>  
72 </p>  
73 </div>  
74 <div class="p-2 px-5">  
75 <p>  
76 <span class="font-semibold">  
77 Cédula:  
78 </span>  
79 <?php echo Sempleado['cedula']; ?>  
80 </p>  
81 </div>  
82 <div class="p-2 px-5">  
83 <p>  
84 <span class="font-semibold">  
85 Fecha de nacimiento:  
86 </span>  
87 <?php echo Sempleado['Fecha_Nac']; ?>  
88 </p>  
89 </div>  
90 <div class="p-2 px-5">  
91 <p>  
92 <span class="font-semibold">  
93 Sexo:  
94 </span>  
95 <?php echo Sempleado['Sexo']; ?>  
96 </p>  
97 </div>  
98 <div class="p-2 px-5">  
99 <p>  
100 <span class="font-semibold">  
101 Correo:  
102 </span>  
103 <?php echo Scontacto['correo']; ?>  
104 </p>  
105 </div>  
16:36:36, 6K, Line 51, Column 37  
main 7 Tab Size: 2 PHP
```

Imprime un titulo mostrando a quien pertenecen los datos, seguido de la foto subida asociada al empleado, para luego empezar a mostrar los datos del empleado, uno por uno.

Datos a mostrar			
Dato	Obrero	Docente	Administrativo
Ruta a la imagen del empleado	✓	✓	✓
Datos del título	✓	✓	✓
Nombre y Apellido	✓	✓	✓
Cédula	✓	✓	✓
Fecha de nacimiento	✓	✓	✓
Sexo	✓	✓	✓

Manual técnico del sistema.



Correo	✓	✓	✓
Municipio en que vive	✓	✓	✓
Parroquia en que vive	✓	✓	✓
Dirección en que vive	✓	✓	✓
Número de teléfono principal	✓	✓	✓
Número de teléfono secundario	✓	✓	✓
Número de teléfono auxiliar	✓	✓	✓
Nivel académico	✓	✓	✓
Título obtenido	✓	✓	✓
Mención	✓	✓	✓
Estudio de segundo nivel	✓	✓	✓
Validez de certificado de salud	✓	✓	✓
Validez de tarjeta de vacunación	✓	✓	✓
Fecha de ingreso a nómina	✓	✓	✓
Carga horaria semanal	✓	✓	✓
Carga horaria mensual	✓	✓	✓
Carga horaria anual	✓	✓	✓
Tiempo en nómina	✓	✓	✓
Rol	✓		
Horas de clase semanales		✓	
Área de clases		✓	

En esta tabla se pueden mostrar los datos que son mostrados en pantalla en base al tipo de empleado que se esté consultando

Manual técnico del sistema.

```
228 > <?php if ($_POST['tipo_empleado'] == 0): ?>
237 > <?php elseif ($_POST['tipo_empleado'] == 1): ?>
254 <?php elseif ($_POST['tipo_empleado'] == 2): ?>
255 <?php endif;?>
```

Esta misma separacion se ve en las siguientes condiciones.

```
<form class="inline-block" action="../../editar/paso-1.php" method="post">
  <input type="hidden" name="id_Persona" value="<?php echo $empleado['id_Persona']; ?>
  ">
  <input type="hidden" name="tipo_empleado" value="<?php echo $_POST['tipo_empleado'];
  ?>">
  <button class="boton" type="submit">Editar</button>
</form>
<form class="inline-block" action="../../controladores/control-empleados.php" method="
post">
  <input type="hidden" name="id_Persona" value="<?php echo $empleado['id_Persona']; ?>
  ">
  <input type="hidden" name="eliminar" value="eliminar">
  <input type="hidden" name="orden" value="eliminar">
  <button class="boton" type="submit">Eliminar</button>
</form>
```

Por último, también se muestran dos botones en pantalla que permiten editar o eliminar al empleado de la base de datos.



Manual técnico del sistema.

Registrar

Consta de un formulario de dos partes.

Paso 1

Vista principal del paso 1.

Nombre del campo	Tipo	Valores / Opciones	Reglas	
nombre	text		Longitud máxima	20
apellido	text		Longitud máxima	20
T_cedula	select	Venezolano(a) Extranjero(a)		
cedula	text		Longitud máxima	8
			Solo dígitos	

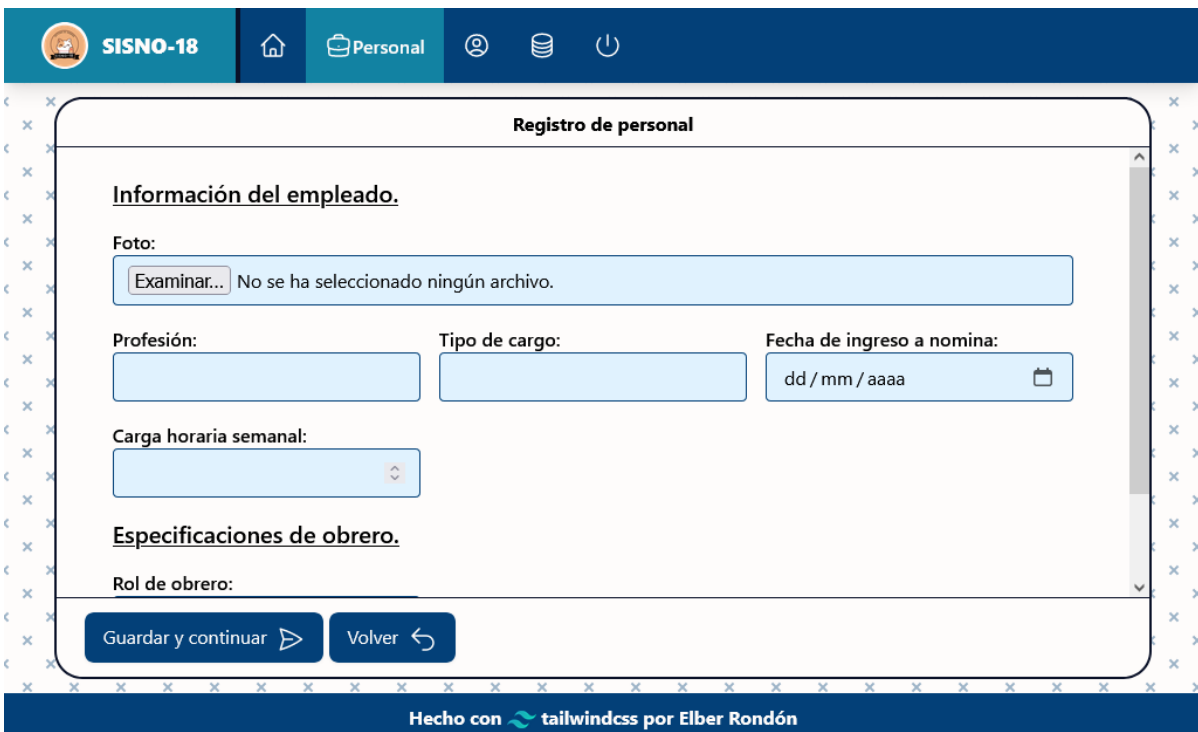
F_nac	date		Rango de fechas	La fecha actual menos 18 años
sexo	radio	Femenino		
		Masculino		
municipio	text			
parroquia	text			
direccion	text			
correo	email		Correo electrónico valido	
pref_P	tel		Longitud máxima	4
			Solo dígitos	
tel_P	tel		Longitud máxima	7
			Solo dígitos	
pref_S	tel		Longitud máxima	4
			Solo dígitos	
tel_S	tel		Longitud máxima	7
			Solo dígitos	
pref_A	tel		Longitud máxima	4
			Solo dígitos	
tel_A	tel		Longitud máxima	7
			Solo dígitos	
N_academico	select	Sin estudios		
		Primaria		
		Bachillerato		
		Universidad		

Manual técnico del sistema.

titulo	text			
mencion	text			
E_2do_nivel	text			
vacuna	text			
dosis_vacuna	number	0	Valor minimo	Mayor o igual a 0
T_empleado	select			
paso-1	hidden	paso-1		

Los datos mostrados en esta tabla son los que el usuario envía al momento de presionar el botón de guardar y continuar.

Paso 2



Registro de personal

Información del empleado.

Foto:

Profesión:


Tipo de cargo:

Fecha de ingreso a nomina:

Carga horaria semanal:

Especificaciones de obrero.

Rol de obrero:

Hecho con  tailwindcss por Elber Rondón

Vista del segundo paso del formulario de registro.

Nombre del campo	Tipo	Valores / Opciones	Reglas
foto	file		Debe ser un archivo jpg, png o gif
profesion	text		
T_cargo	text		
FI_nomina	date		
horas	number		Debe ser mayor a 6 horas
rol	text		
nombre	hidden		Recibido del paso anterior
apellido	hidden		Recibido del paso anterior
cedula	hidden		Recibido del paso anterior
T_empleado	hidden		Recibido del paso anterior
F_nac	hidden		Recibido del paso anterior
sexo	hidden		Recibido del paso anterior
sexo	hidden		Recibido del paso anterior
municipio	hidden		Recibido del paso anterior
parroquia	hidden		Recibido del paso anterior
direccion	hidden		Recibido del paso anterior
N_academico	hidden		Recibido del paso anterior
titulo	hidden		Recibido del paso anterior
encion	hidden		Recibido del paso anterior
E_2do_nivel	hidden		Recibido del paso anterior



Manual técnico del sistema.

correo	hidden		Recibido del paso anterior
pref_P	hidden		Recibido del paso anterior
tel_P	hidden		Recibido del paso anterior
pref_S	hidden		Recibido del paso anterior
tel_S	hidden		Recibido del paso anterior
pref_A	hidden		Recibido del paso anterior
tel_A	hidden		Recibido del paso anterior
vacuna	hidden		Recibido del paso anterior
dosis_vacuna	hidden		Recibido del paso anterior
paso-2	hidden	paso-2	
orden	hidden	insertar	

Una vez que el usuario ingrese los datos del nuevo empleado, estos serán enviados al controlador `contro-empleados.php` para su ingreso en la base de datos.



Manual técnico del sistema.

Editar

Tómalos datos de un empleado para generar una vista de edición con los datos almacenados en la base de datos

SISNO-18 Personal

Registro de personal

Información personal.

Nombre: Morganica

Apellido: Lawty

Cédula: V12649473

Fecha de nacimiento: 14 / 04 / 2022

Cédula no editable.

Sexo:

Femenino ☐ Masculino ☒

Información de contacto.

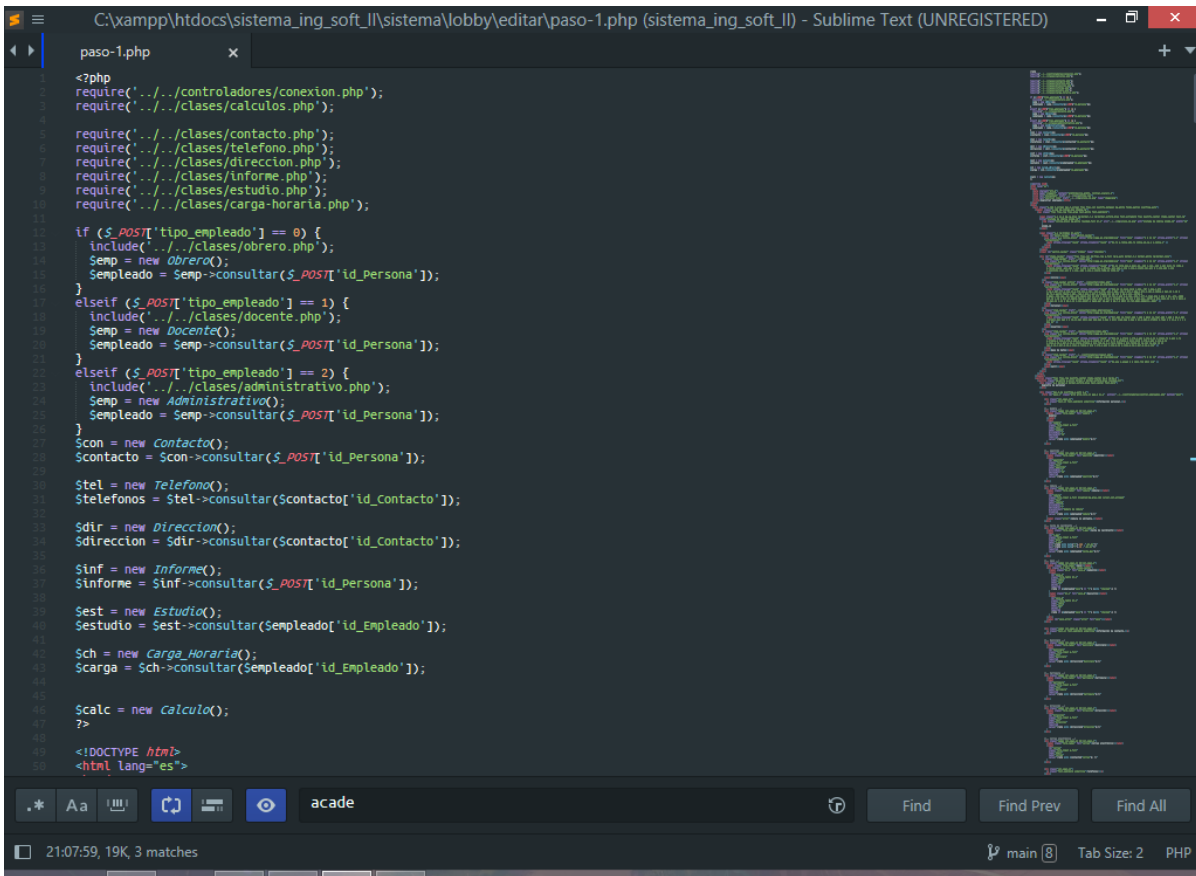
Municipio: Provincia: Dirección:

Guardar cambios Volver

Hecho con tailwindcss por Elber Rondón

Vista general del formulario de edición.

Manual técnico del sistema.



Inicia requiriendo los archivos de clase necesarios, así como el controlador de conexión a base de datos, posteriormente evalúa en base al tipo de empleado para requerir e instanciar las clases Obrero, Docente y Administrativo.

De igual manera, instancia las demás clases y hace las consultas necesarias.

Una vez hecho esto, se imprimen los valores obtenidos en la consulta como contenido de los campos del formulario de edición.

Nombre del campo	Tipo	Valores / Opciones	Reglas	
nombre	text		Longitud Máxima	20
apellido	text		Longitud Máxima	20

F_nac	date	14-04-2022		
sexo	radio	F		
		M		
municipio	text			
parroquia	text			
direccion	text			
correo	email			
pref_P	tel		Longitud Máxima	4
tel_P	tel		Longitud Máxima	7
pref_S	tel		Longitud Máxima	4
tel_S	tel		Longitud Máxima	7
pref_A	tel		Longitud Máxima	4
tel_A	tel		Longitud Máxima	7
N_academico	select	Sin estudios		
		Primaria		
		Bachillerato		
		Universidad		
titulo	text			
mencion	text			
E_2do_nivel	text			



Manual técnico del sistema.

FI_nomina	date		
horas	number		
rol	text		
id_Persona	hidden	Id obtenido desde base de datos	
T_empleado	hidden		
editar	hidden	editar	
orden	hidden	editar	

Cabe mencionar que tanto la cédula como la foto del empleado no son editables.

Por último, los datos son enviados a control-empleados.php para actualizar el registro en la base de datos.