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

**Sumario del manual**

[Presentación. 5](#_Toc127125341)

[Resumen. 5](#_Toc127125342)

[Finalidad Del Manual. 6](#_Toc127125343)

[Introducción. 6](#_Toc127125344)

[Aspectos teóricos 6](#_Toc127125345)

[Diagramas De modelamiento 6](#_Toc127125346)

[Aspecto técnico del desarrollo del sistema 6](#_Toc127125347)

[Requerimientos del software 6](#_Toc127125348)

[Aspectos técnicos. 7](#_Toc127125349)

[Herramientas utilizadas para el desarrollo. 7](#_Toc127125350)

[SublimeText 7](#_Toc127125351)

[GitHub 7](#_Toc127125352)

[Node.js 8](#_Toc127125353)

[TailwindCSS. 8](#_Toc127125354)

[Xampp 9](#_Toc127125355)

[Diagramas de modelado. 10](#_Toc127125356)

[Modelo entidad relacion. 10](#_Toc127125357)

[Diccionario de datos. 11](#_Toc127125358)

[administrativos. 11](#_Toc127125359)

[carga\_horaria 11](#_Toc127125360)

[contactos 12](#_Toc127125361)

[coordinacion 12](#_Toc127125362)

[direcciones 12](#_Toc127125363)

[directivo 13](#_Toc127125364)

[docentes 13](#_Toc127125365)

[empleados 14](#_Toc127125366)

[estudios 14](#_Toc127125367)

[informes 15](#_Toc127125368)

[obreros 15](#_Toc127125369)

[personal\_doc 15](#_Toc127125370)

[personas 16](#_Toc127125371)

[telefonos 16](#_Toc127125372)

[usuarios 17](#_Toc127125373)

[Requerimientos del software. 18](#_Toc127125374)

[Requisitos mínimos. 18](#_Toc127125375)

[Especificaciones dentro del código. 19](#_Toc127125376)

[Estructura de directorios 19](#_Toc127125377)

[Clases. 21](#_Toc127125378)

[administrativo.php 22](#_Toc127125379)

[Descripción de los métodos. 23](#_Toc127125380)

[carga-horaria.php 27](#_Toc127125381)

[Descripción de los métodos 28](#_Toc127125382)

[contacto.php 30](#_Toc127125383)

[Descripción de los métodos 31](#_Toc127125384)

[direccion.php 34](#_Toc127125385)

[Descripción detallada de los métodos 35](#_Toc127125386)

[docente.php 38](#_Toc127125387)

[Descripción de los métodos 39](#_Toc127125388)

[empleado.php 43](#_Toc127125389)

[Descripción de los métodos 44](#_Toc127125390)

[estudio.php 49](#_Toc127125391)

[Descripción de los métodos 50](#_Toc127125392)

[informe.php 53](#_Toc127125393)

[Descripción de los métodos 54](#_Toc127125394)

[obrero.php 57](#_Toc127125395)

[Descripción de los métodos. 58](#_Toc127125396)

[persona.php 62](#_Toc127125397)

[Descripción de los métodos 63](#_Toc127125398)

[telefono.php 66](#_Toc127125399)

[Descripción de los métodos 67](#_Toc127125400)

[usuario.php 70](#_Toc127125401)

[Descripción de los métodos 71](#_Toc127125402)

[Controladores 75](#_Toc127125403)

[base-datos.php 75](#_Toc127125404)

[conexion.php 77](#_Toc127125405)

[control-empleados.php 79](#_Toc127125406)

[docente-excel.php 89](#_Toc127125407)

[docente-pdf.php 92](#_Toc127125408)

[login.php 95](#_Toc127125409)

[logout.php 97](#_Toc127125410)

[Vistas o pantalla 98](#_Toc127125411)

[Diagrama general del sistema 98](#_Toc127125412)

[Login 99](#_Toc127125413)

[Menú principal. 103](#_Toc127125414)

[Consultar. 107](#_Toc127125415)

[Consulta de personal 110](#_Toc127125416)

[Consulta de obreros 113](#_Toc127125417)

[Consulta de docentes 116](#_Toc127125418)

[Consulta de personal administrativo 119](#_Toc127125419)

[Consulta de usuarios 122](#_Toc127125420)

[Consultar empleado 125](#_Toc127125421)

[Registrar 130](#_Toc127125422)

[Paso 1 130](#_Toc127125423)

[Paso 2 132](#_Toc127125424)

[Editar 135](#_Toc127125425)

**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.

# 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.

# 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; aí como tambien puede llevar seguimiento de los cambios.

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.

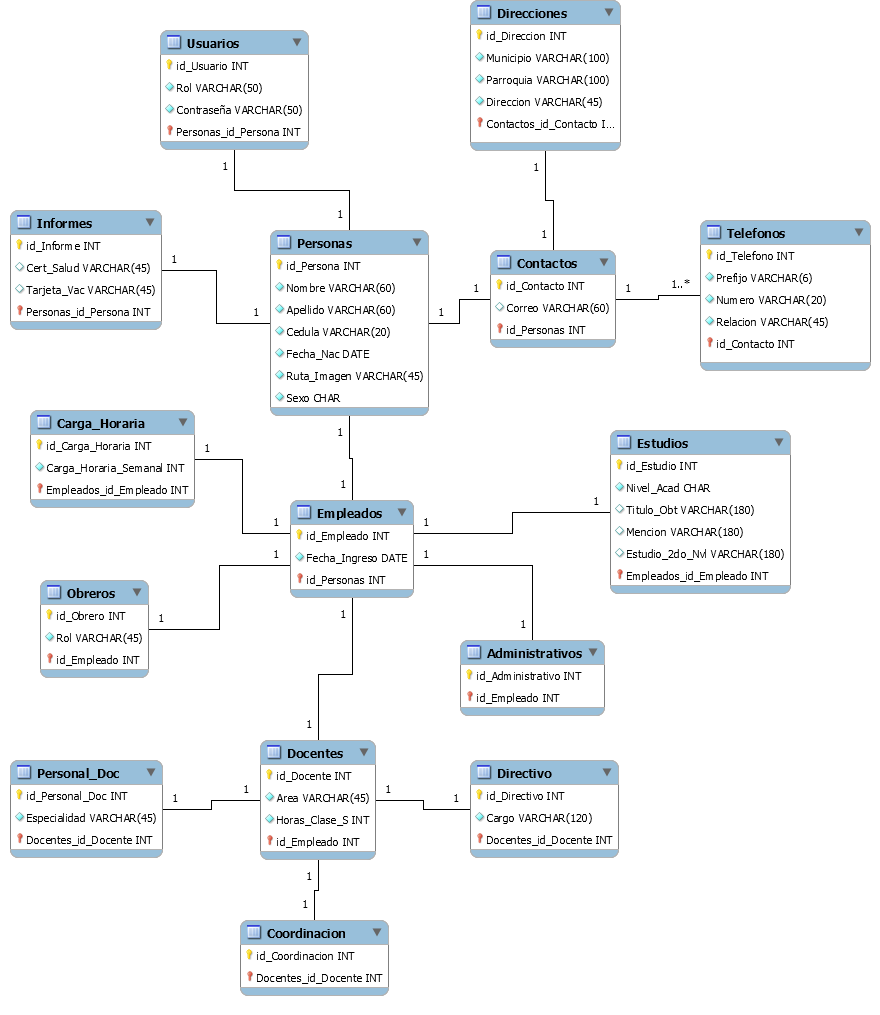
Fue usado para el maquetado del proyecto. Gracias a tailwind CLI, el cual permite, en conjunto con NPM de Node.js, identificar que clases del framework se están usando para generar un compilado que incluya solo las clases usadas para mantener cortos los tiempos de carga en conexiones lentas.

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.



### 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 *(Primaria)* | int(11) | No |  |  |
| id\_Empleado *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Carga\_Horaria\_Semanal | int(11) | No |  |  |
| Empleados\_id\_Empleado *(Primaria)* | 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 |

contactos**.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo** | **Nulo** | **Predeterminado** | **Enlaces a** |
| id\_Contacto *(Primaria)* | int(11) | No |  |  |
| Correo | varchar(60) | Sí | *NULL* |  |
| id\_Personas *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Docentes\_id\_Docente *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Municipio | varchar(100) | No |  |  |
| Parroquia | varchar(100) | No |  |  |
| Direccion | varchar(45) | No |  |  |
| Contactos\_id\_Contacto *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Cargo | varchar(120) | No |  |  |
| Docentes\_id\_Docente *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Area | varchar(45) | No |  |  |
| Horas\_Clase\_S | int(11) | No |  |  |
| id\_Empleado *(Primaria)* | 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 |

empleados**.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo** | **Nulo** | **Predeterminado** | **Enlaces a** |
| id\_Empleado *(Primaria)* | int(11) | No |  |  |
| Fecha\_Ingreso | date | No |  |  |
| id\_Personas *(Primaria)* | 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 *(Primaria)* | 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 *(Primaria)* | 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 |

informes**.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo** | **Nulo** | **Predeterminado** | **Enlaces a** |
| id\_Informe *(Primaria)* | int(11) | No |  |  |
| Cert\_Salud | varchar(45) | Sí | *NULL* |  |
| Tarjeta\_Vac | varchar(45) | Sí | *NULL* |  |
| Personas\_id\_Persona *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Rol | varchar(45) | No |  |  |
| id\_Empleado *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Especialidad | varchar(45) | No |  |  |
| Docentes\_id\_Docente *(Primaria)* | 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 *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Prefijo | varchar(6) | No |  |  |
| Numero | varchar(20) | No |  |  |
| Relacion | varchar(45) | No |  |  |
| id\_Contacto *(Primaria)* | 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 *(Primaria)* | int(11) | No |  |  |
| Rol | varchar(50) | No |  |  |
| Contraseña | varchar(50) | No |  |  |
| Personas\_id\_Persona *(Primaria)* | 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 |

# 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

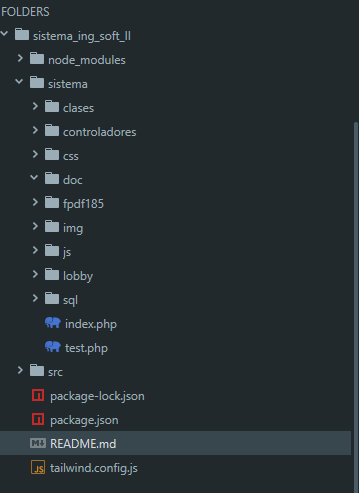
• Memoria RAM: 1GB

• Disco Duro: 1GB

# 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.
* **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: i**ncluye 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.

## 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.

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. |

#### Descripción de los métodos.

**insertarAdministrativo:**

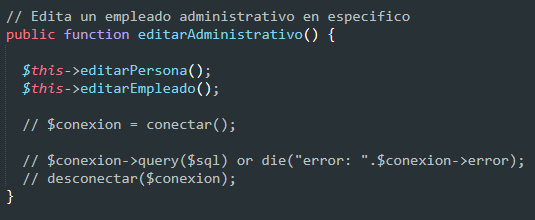


Ejecuta los métodos heredados insertarPersona e insertarEmpleado. Luego realiza una conexión con la base de datos, Se genera una variable local con el id del empleado (debe haberse establecido el atributo antes de ejecutarlo).

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.

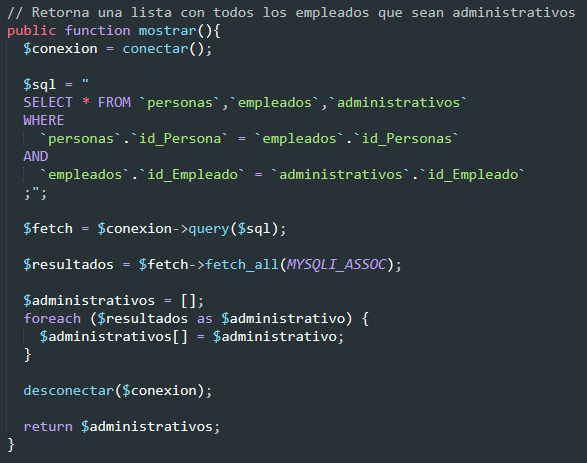
**editarAdministrativo:**



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:**

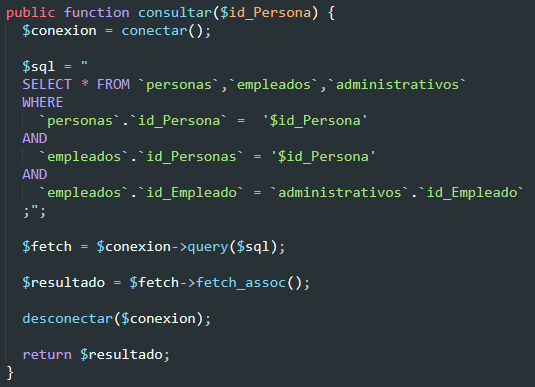


Este método usa polimorfismo, y se diferencia del método presente en su clase padre.

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 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.**



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

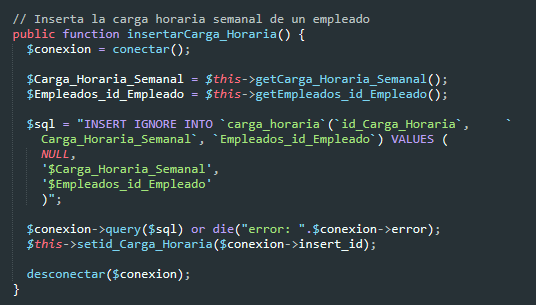
Se ejecutará la consulta y traerá los datos del empleado como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos del empleado.

### 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. |

Descripción de los métodos**:**

**insertarCarga\_Horaria:**

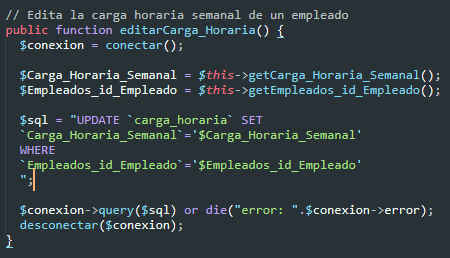


Comienza iniciando una conexión a base de datos, luego guarda en variables locales la carga horaria semanal y el id de empleado.

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.

**editarCarga\_Horaria:**

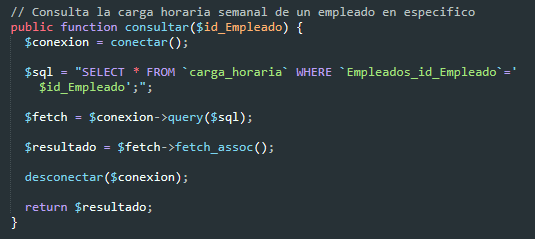


Comienza iniciando una conexión a base de datos, luego guarda en variables locales la carga horaria semanal y el id de empleado.

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:**



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.

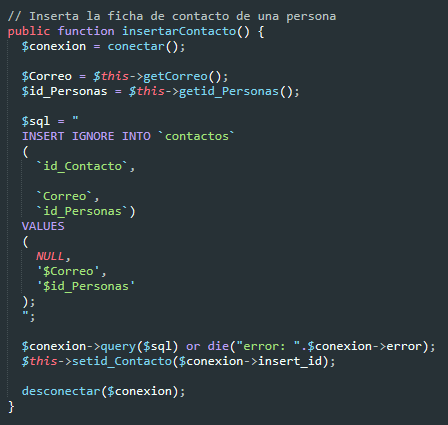
Finalmente cierra la conexión a base de datos.

### 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 |

Descripción de los métodos**.**

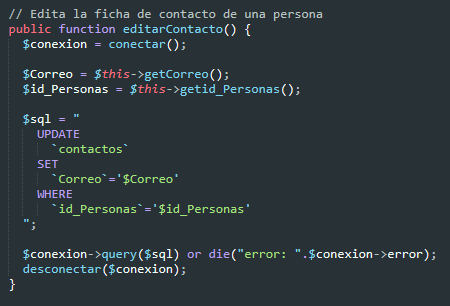
**insertarContacto:**



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.

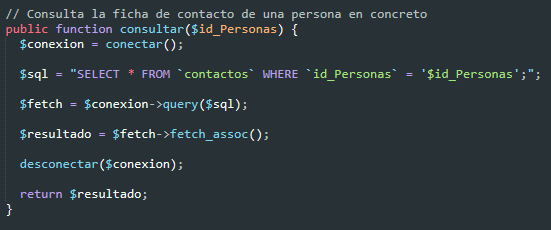
**editarContacto:**



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:**



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.

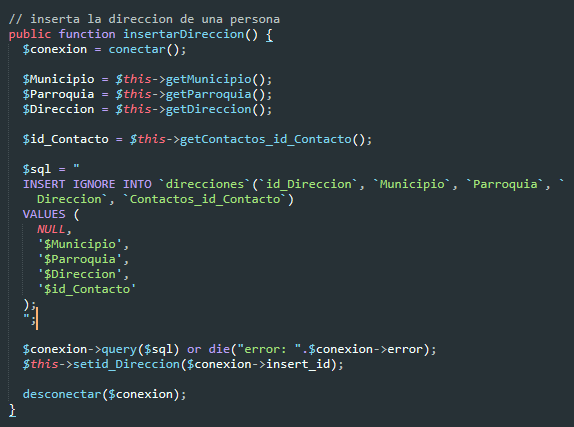
Entonces, traerá lo datos de la ficha de contacto como un arreglo asociativo. Al finalizar, cierra la conexión a base de datos y retorna los datos obtenidos.

### 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. |

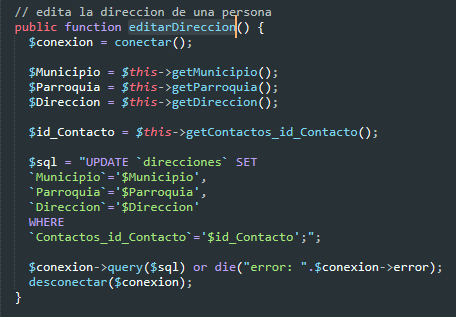
Descripción detallada de los métodos**.**

**insertarDireccion:**



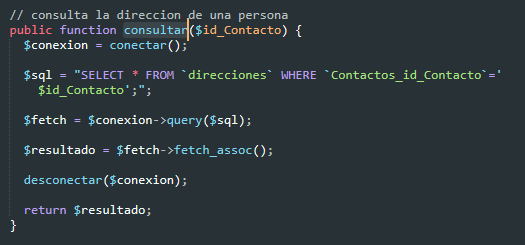
El método almacena en variables locales el municipio, parroquia, dirección e id de la ficha de contacto del empleado, prepara la sentencia SQL para insertar esos datos y la ejecuta, de haber un error se mostrará, finalmente almacena como atributo el id de la dirección insertada y cierra la conexión con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

**editarDireccion:**



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:**



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. |

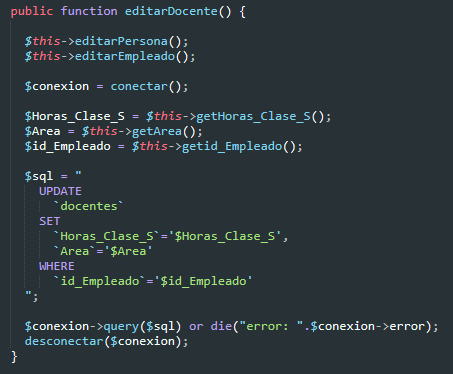
Descripción de los métodos**:**

**insertarDocente:**



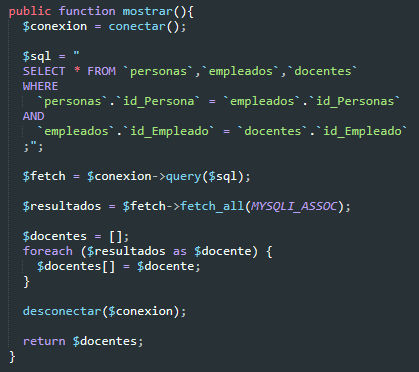
El método inicia utilizando los métodos heredados insertarPersona e insertarEmpleado, luego inicia una conexión a base de datos, almacena en variables locales las horas de clase impartidas por el docente, su área e id de empleado, prepara la sentencia SQL para insertar, luego la ejecuta o muestra un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

**editarDocente:**



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.

**mostrar:**

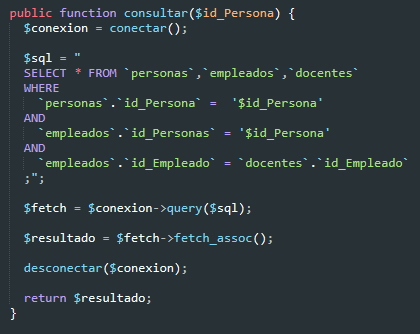


Este método usa polimorfismo, y se diferencia del método presente en su clase padre.

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.

**consultar:**



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.

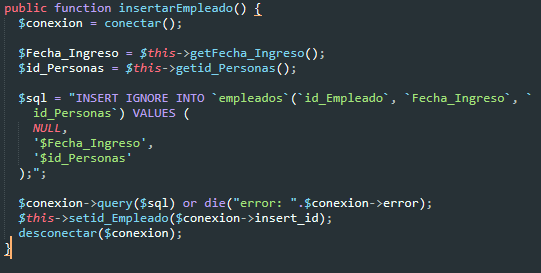
Se ejecutará la consulta y traerá los datos del empleado como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos del empleado.

### 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. |

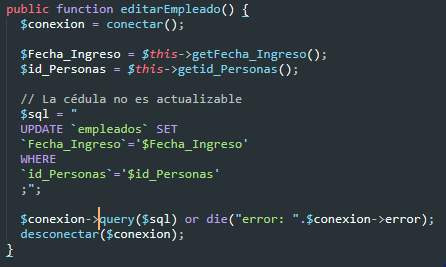
Descripción de los métodos**.**

**insertarEmpleado:**



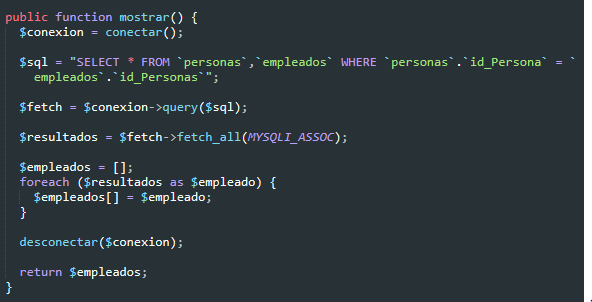
El método inicia una conexión con la base de datos, luego almacena la fecha de ingreso a nómina y el id de persona en variables locales, luego prepara la sentencia SQL para insertar, la ejecuta y muestra un mensaje de error en caso de fallar. Finalmente, guarda como atributo el id del empleado registrado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

**editarEmpleado:**



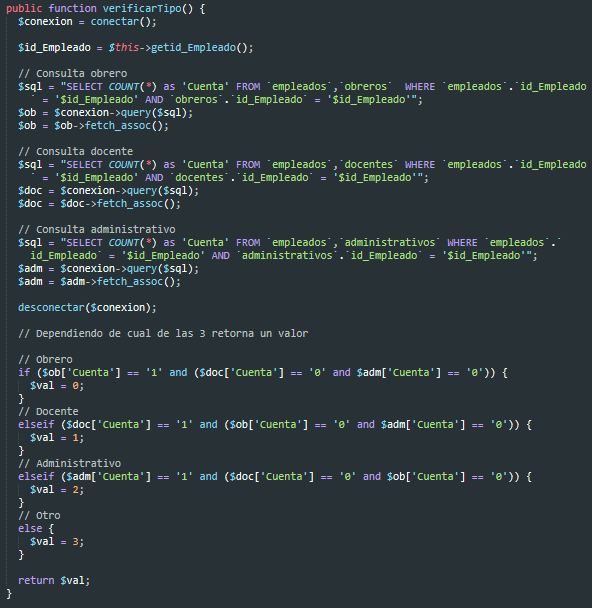
El método inicia una conexión con la base de datos, luego almacena la fecha de ingreso a nómina y el id de persona en variables locales, luego prepara la sentencia SQL para editar los valores actuales, la ejecuta y muestra un mensaje de error en caso de fallar. Finalmente, cierra la conexión a base de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

**mostrar:**



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.

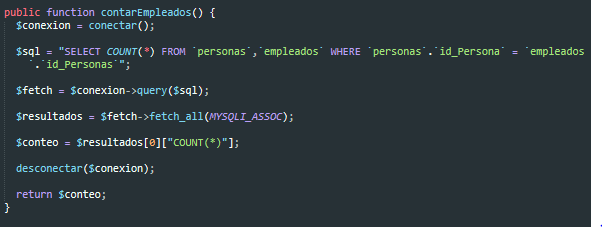
**verificarTipo:**



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.

**contarEmpleados:**



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 |

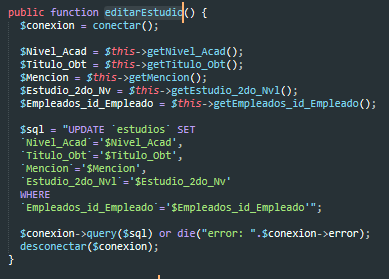
Descripción de los métodos**.**

**insertarEstudio:**



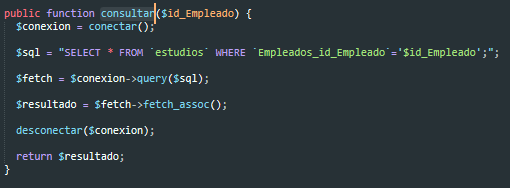
El método inicia una conexión a la base de datos, guarda el nivel académico, título académico, la mención, el estudio de segundo nivel y el id de empleado en variables locales. Luego prepara la instrucción SQL para insertar los valores, la ejecuta y muestra un mensaje de error en caso de fallar, finalmente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

**editarEstudio:**



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.

**consultar:**



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.

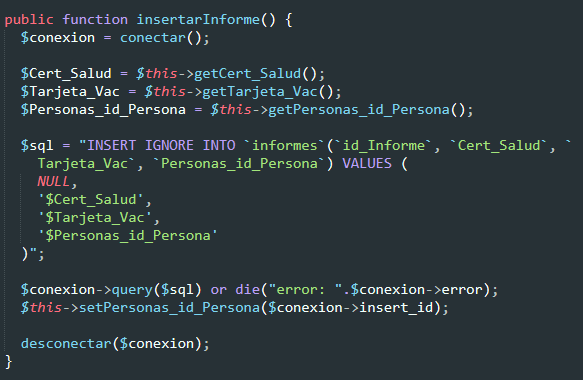
Se ejecutará la consulta y traerá los datos de estudios como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos de la base de datos.

### 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. |

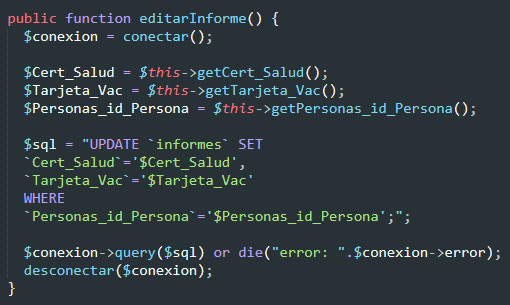
Descripción de los métodos**:**

**insertarInforme:**



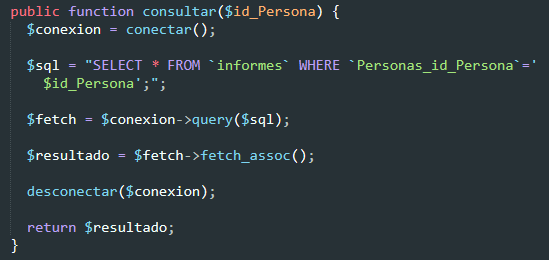
El método inicia una conexión a la base de datos, guarda los valores del certificado de salud y el de la tarjeta de vacunación. Luego prepara la instrucción SQL para insertar los valores, la ejecuta y muestra un mensaje de error en caso de fallar, finalmente guarda como atributo el id del registro realizado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

**editarInforme:**



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:**



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.

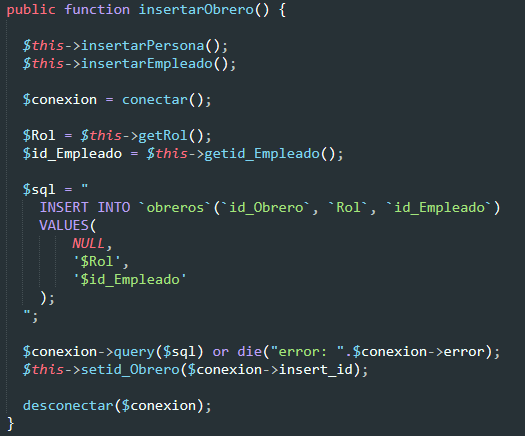
Se ejecutará la consulta y traerá los datos del informe médico como un arreglo. Cerrará la conexión a base de datos y retornará los datos traídos de la base de datos.

### 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 |

#### Descripción de los métodos.

**insertarObrero:**



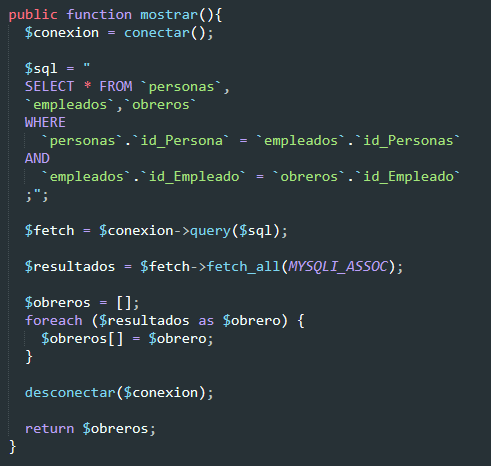
El método inicia ejecutando los métodos heredados insertarPersona e insertarEmpleado, luego inicia una conexión a base de datos y guarda el rol e id del empleado en variables locales, prepara la sentencia SQL para insertar. Ejecuta la consulta y lanza un mensaje de error en caso de fallar, almacena como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

**editarObrero:**



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.

**mostrar:**

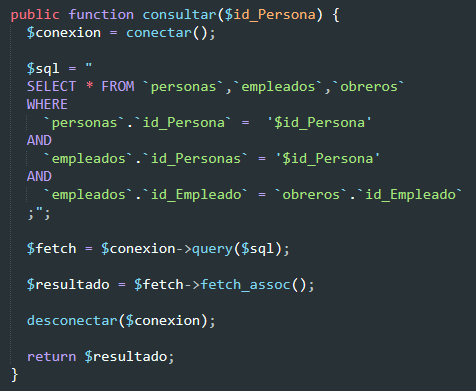


Este método usa polimorfismo, y se diferencia del método presente en su clase padre.

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.

**consultar:**



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.

### 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. |

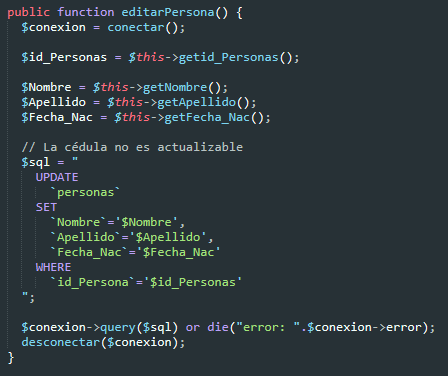
Descripción de los métodos**.**

**insertarPersona:**



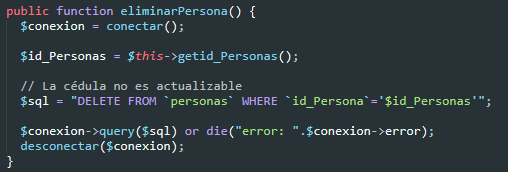
El método inicia una conexión a base de datos, luego almacena nombre, apellido, cédula, fecha de nacimiento, ruta en que almacena la imagen de la persona y su sexo en variables locales. Prepara la sentencia SQL para insertar, la ejecuta o muestra un mensaje de error en caso de fallar, finalmente, almacena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

**editarPersona:**



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.

**eliminarPersona:**



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.

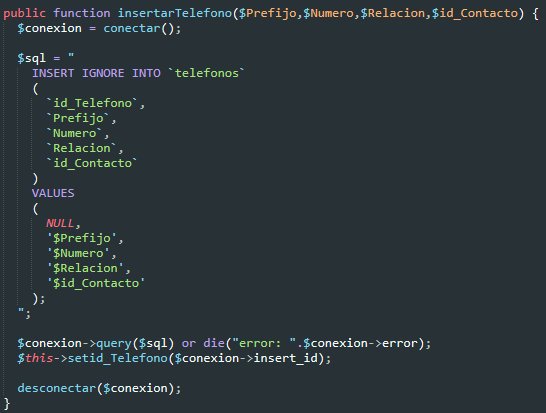
Al eliminar a una persona se estarán eliminando los registros que estén asociados a esta en cascada.

### 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. |

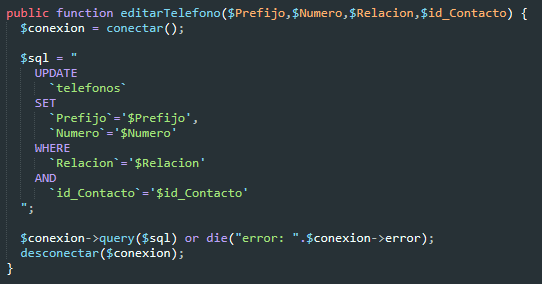
Descripción de los métodos.

**insertarTelefono:**



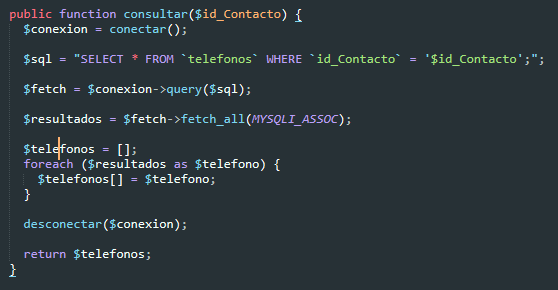
El método inicia una conexión a la base de datos y prepara la sentencia SQL para insertar usando como valores los parámetros recibidos, luego la ejecuta y en caso de fallar mostrará un mensaje de error, finalmente almacena como atributo el id del teléfono registrado y cierra la conexión con la base de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

**editarTelefono:**



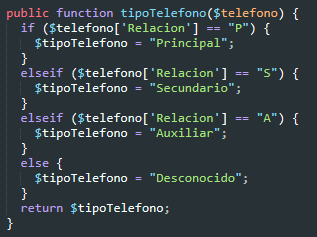
El método inicia una conexión a la base de datos y prepara la sentencia SQL para editar usando como valores los parámetros recibidos, luego la ejecuta y en caso de fallar mostrará un mensaje de error, finalmente cierra la conexión con la base de datos.

**consultar:**



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 obtenido como un arreglo de arreglos asociativos.

**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:**



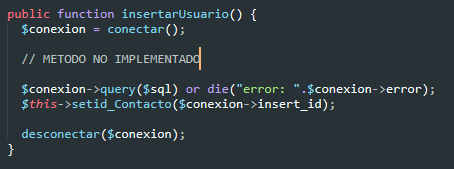
El método concatena el prefijo y el número telefónico y lo retorna.

### 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. |

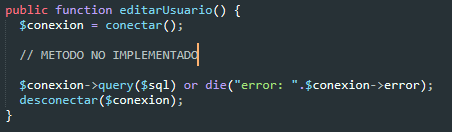
Descripción de los métodos**.**

**insertarUsuario:**



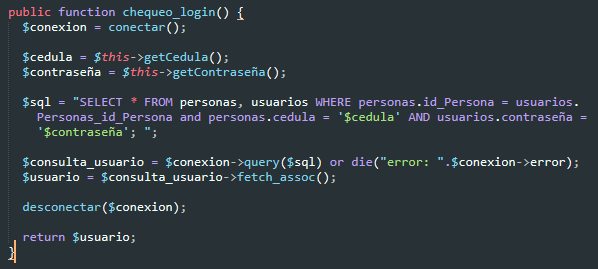
El método no se encuentra implementado de una manera funcional por los momentos.

**editarUsuario:**



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

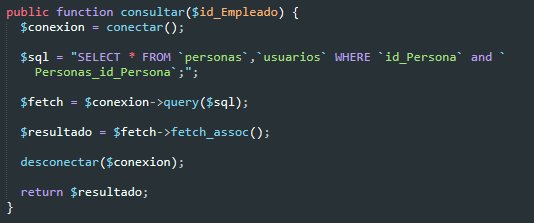
**chequeo\_login:**



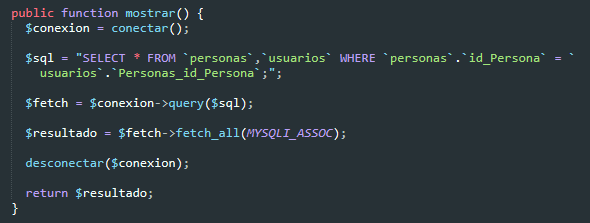
El método inicia una conexión a la base de datos, guarda la cedula y la contraseña del usuario en variables locales y prepara la sentencia SQL, esta misma busca la existencia de una persona con la cedula especificada que sea usuario y posea la contraseña especificada.

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

**consultar:**

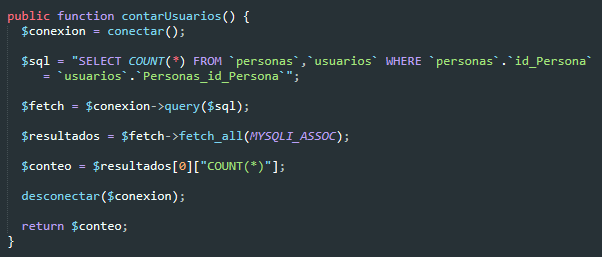


**mostrar:**



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**

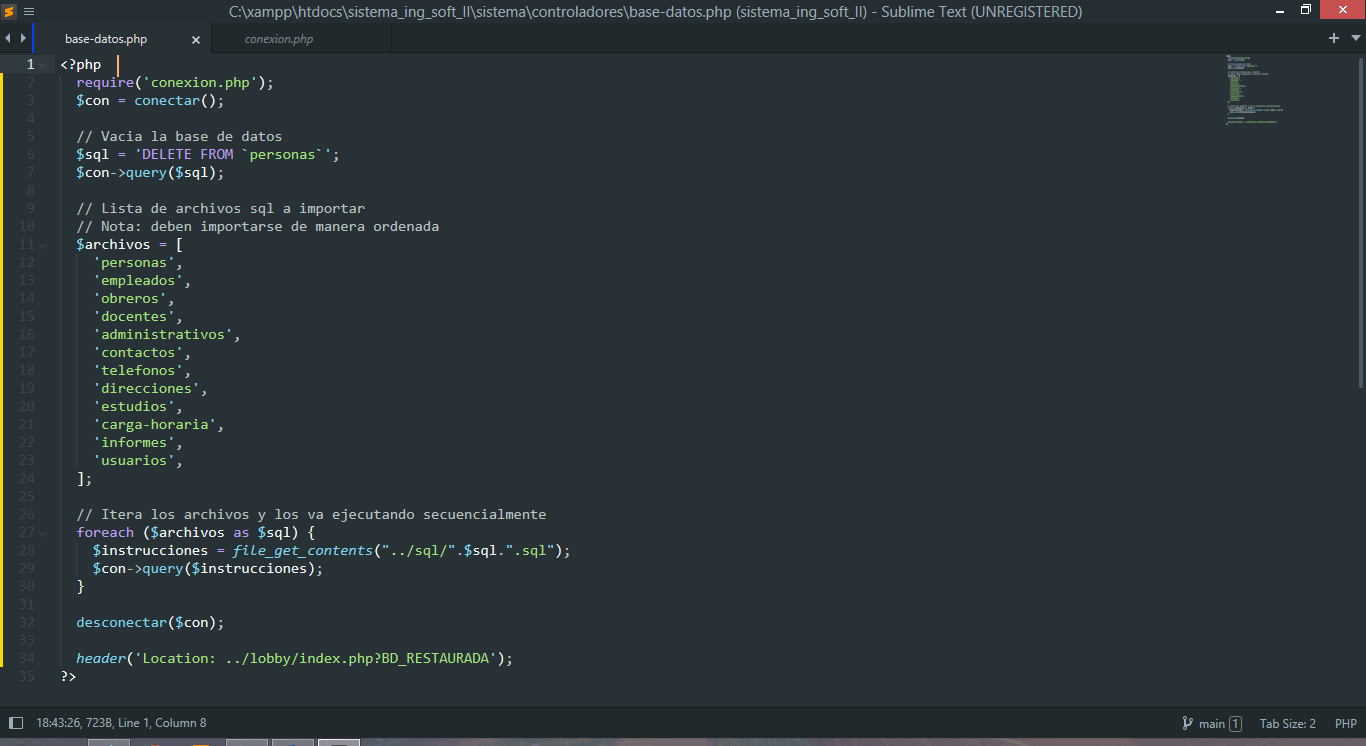


El método se conecta con la base de datos y prepara una sentencia SQL para consultar el número de usuarios existentes, la ejecuta o muestra un mensaje de error en caso de fallar, cierra la conexión y retorna el conteo de usuarios registrados.

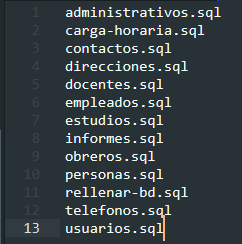
Controladores**.**

### base-datos.php

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



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:



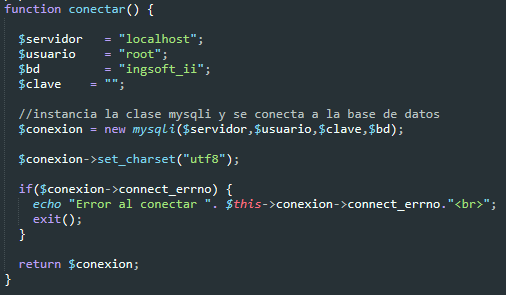
Entonces entrará en un bucle foreach en que usará la función *file\_get\_contents* para obtener el contenido de un archivo sql como un string que posteriormente será ejecutado con el método *query*  de la conexión instanciada con la base de datos. Hará esto para cada archivo mencionado anteriormente.

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.

### conexion.php

Gestiona la conexión y desconexión con la base de datos.

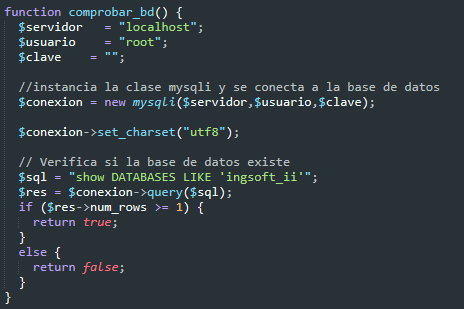
Este controlador posee tres funciones, *conectar*, *comprobar\_bd* y *desconectar.*



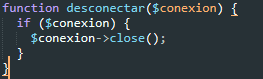
La función conectar cuenta con las variables para instanciar una conexión a base de datos mediante mysqli, se debe especifica el servidor, usuario a usar, el nombre de la base de datos (por defecto ingsoft\_ii) y la contraseña de usuario.

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.



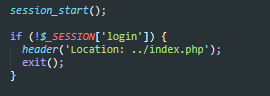
La segunda función, comprobar\_bd, se conecta a base de datos como en la función anterior, con la diferencia de que no se especifica una base de datos. Posterior a eso, consulta si existe la base de datos, si hay al menos una fila (base de datos que coincida) retorna true, por lo contrario retornará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



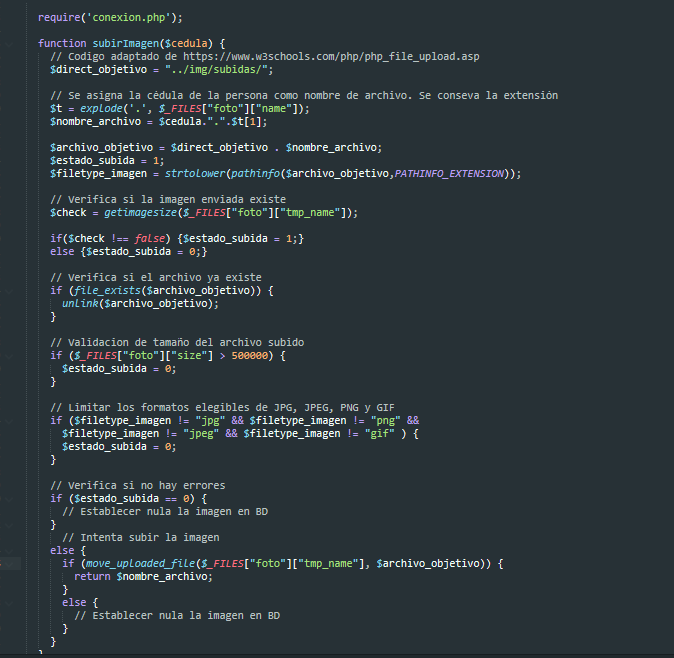
Por último, está la función desconectar. Toma como parámetro la conexión generada por la función conectar. Si esa conexión existe, la cierra.

### 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.



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

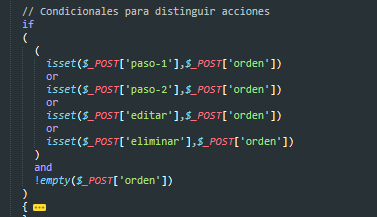


En segundo lugar, este controlador cuenta con una función llamada *subirImagen* la cual recibe la cédula del empleado en cuestión.

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).



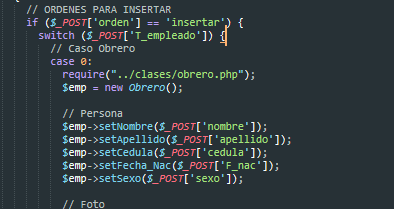
Este controlador cuenta con un condicional que utiliza las variables enviadas a la hora de enviar un formulario. Con el propósito de validar que se envió correctamente el formulario. Así como verificar que se haya enviado una orden a realizar.

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



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

**Caso Insertar:**



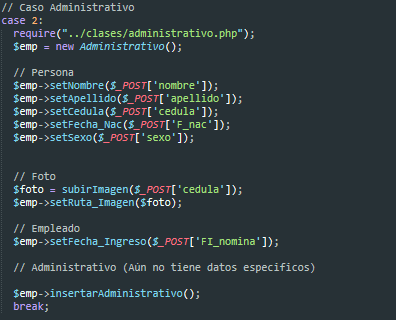
Para el caso de insertar, se tomará en cuenta mediante un switch que tipo de empleado se va a registrar, pudiendo ser obrero, docente o administrativo.



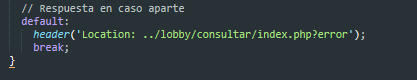
Para el caso de obreros, se instancia un objeto de la clase obrero al que se le asignan sus datos personales, se usa la función subirImagen y se establece la ruta de la imagen subida, luego se agregará la fecha de ingreso a nómina y el rol de obrero que cumple, finalmente se ejecutará el método insertarObrero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



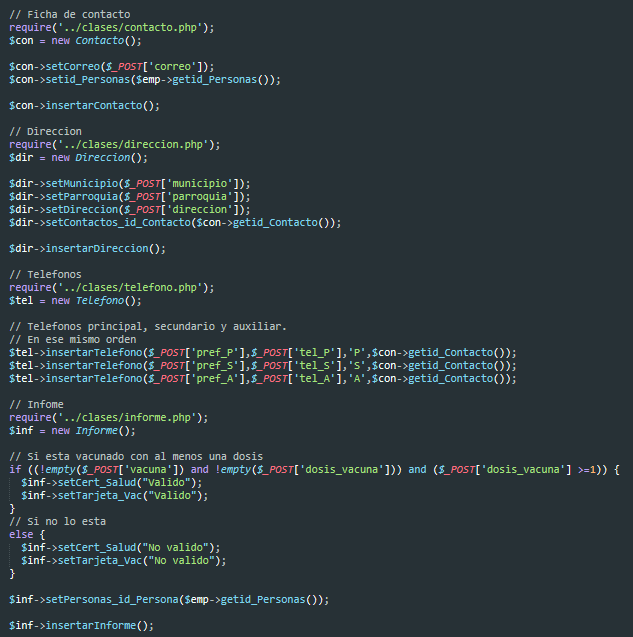
En el caso de ser docente, funcionará de modo similar, instanciará un objeto de la clase Docente, establecerá los datos personales, la imagen y la fecha de ingreso a nómina; se diferencia del anterior dado que se establece el número de horas de clase semanales y el área al que corresponde, finalmente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



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



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

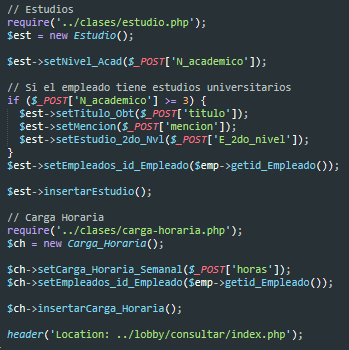


Una vez registrados los datos personales y laborales del empleado, se instanciara las clases Contacto, Telefono e Informe.

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

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.



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.

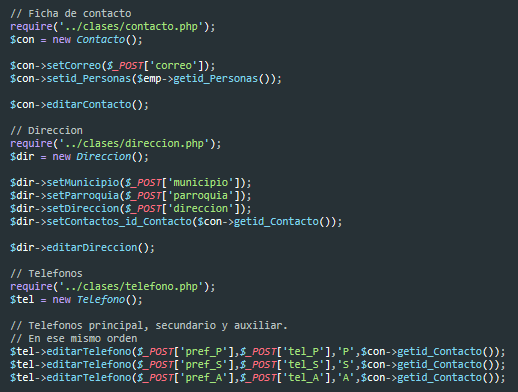
Para Carga\_Horaria, se almacena el número de horas laborales por semana que cumple el empleado, y el id del empleado para luego anexarlo a base de datos.

**Caso editar:**

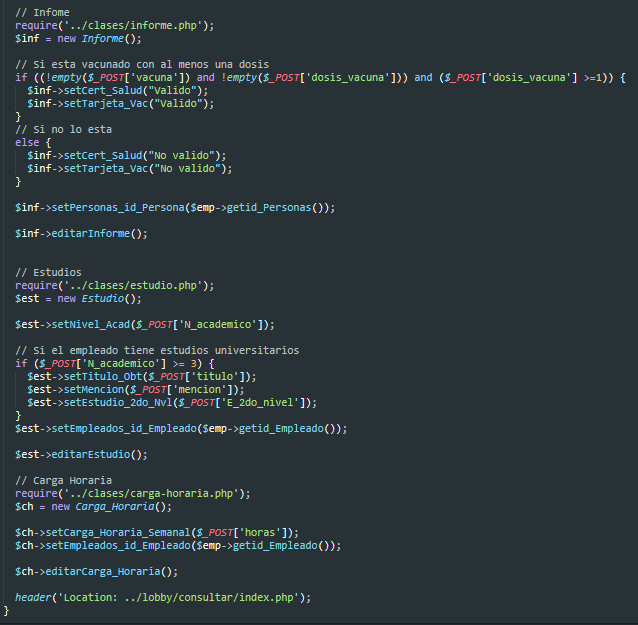
Para el caso de editar, el procedimiento es similar al de insertar, con las diferencias de que al instanciar las clases Obrero, Docente o Administrativo, no se especifica ni número de cédula ni la imagen, estos mismos no son editables.

|  |  |
| --- | --- |
| Caso Obrero | Caso Docente |
|  |  |
| Caso Administrativo | Caso aparte |
|  |  |

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.

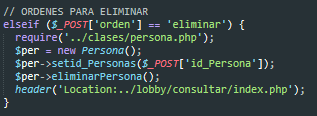


Posteriormente continua con las clases Contacto, Dirección y Telefono. Establece sus datos y se envían a base de datos con sus respectivos métodos para editar.



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

**Caso eliminar:**

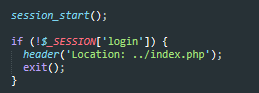


Para el caso eliminar, se instancia la clase persona, se asigna el id de la persona (empleado) y se ejecuta el método eliminarPersona para eliminar su registro de la base de datos.

Posteriormente el usuario es enviado al área de consulta.

### docente-excel.php

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



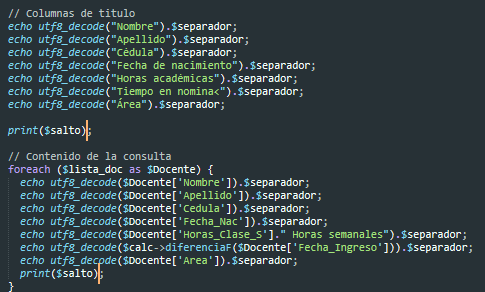
En un comienzo el controlador verifica que el usuario haya iniciado sesión.



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.

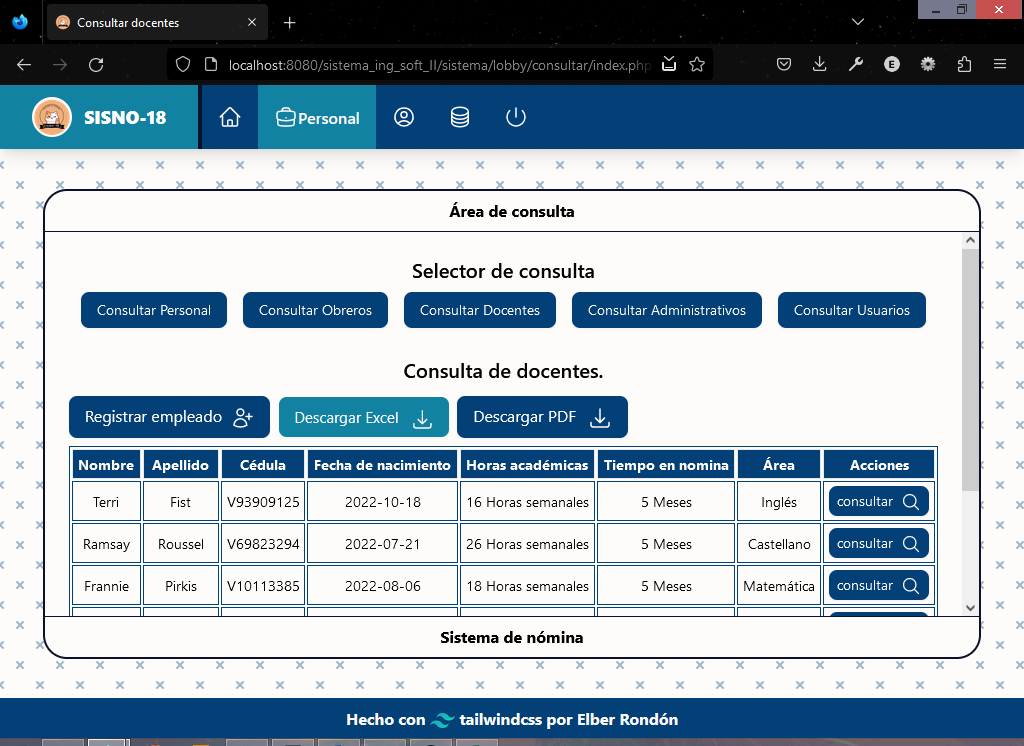
Luego se especifica el nombre del archivo y las variables que especificarán saltos de línea (paso a siguiente fila enfocando la primera columna) y tabulaciones (paso a la siguiente columna a la derecha) al generar el Excel.

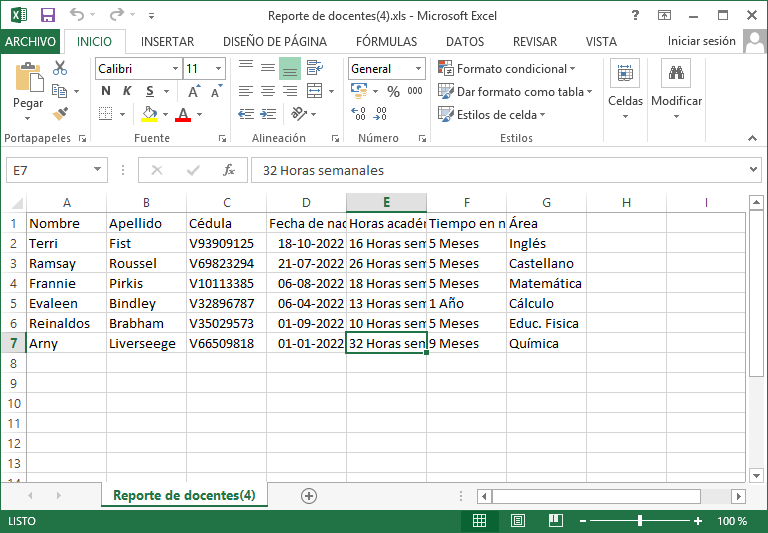


Una vez definido esto, se comienza a formatear el documento Excel al usar echo. Primero la cabecera usando la función utf8\_decode para convertir las cadenas de texto a codificación ISO-8859-1.

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.

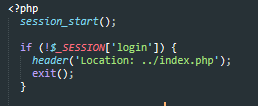
Finalmente el archivo Excel será descargado por el navegador del usuario.



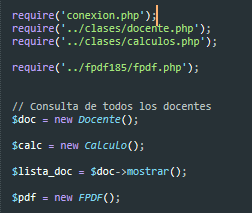


### docente-pdf.php

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

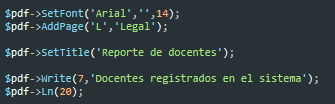


Inicialmente se verifica que la sesión del usuario exista.



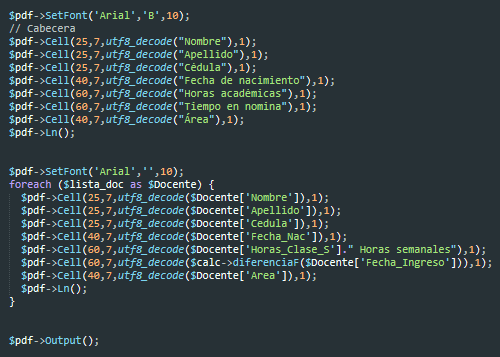
Se llama el controlador de conexión a base de datos, la clase Docente y la interfaz Calculo, además de archivo principal de fpdf.

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.



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).

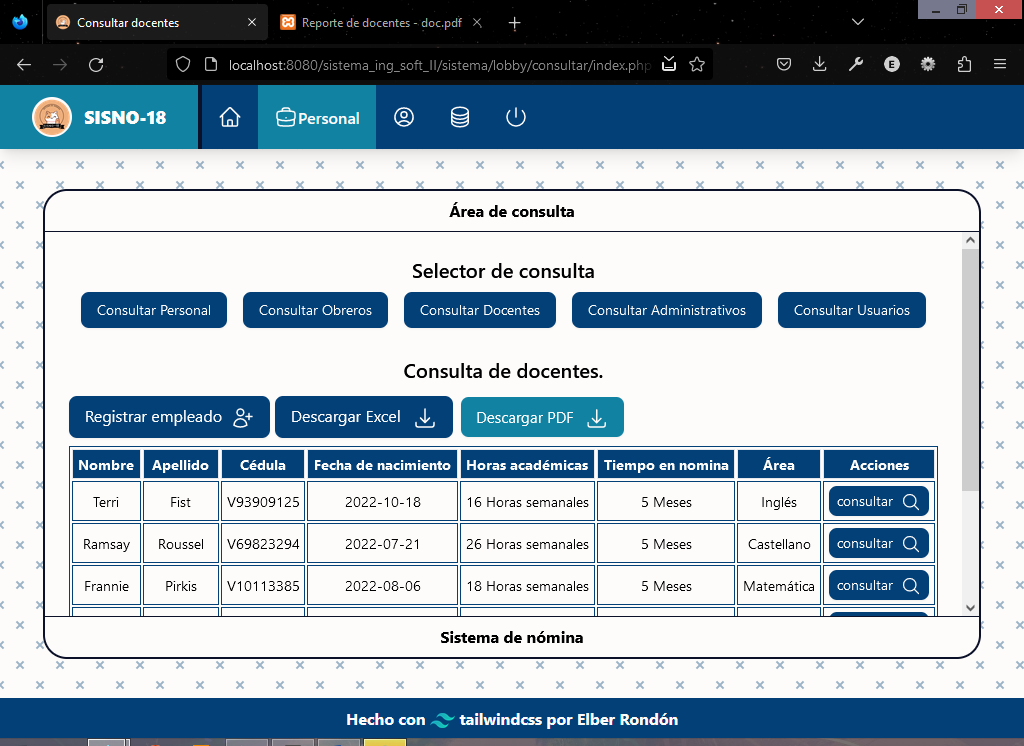
Luego se define el nombre del documento, se inserta el título del documento y un salto de línea.

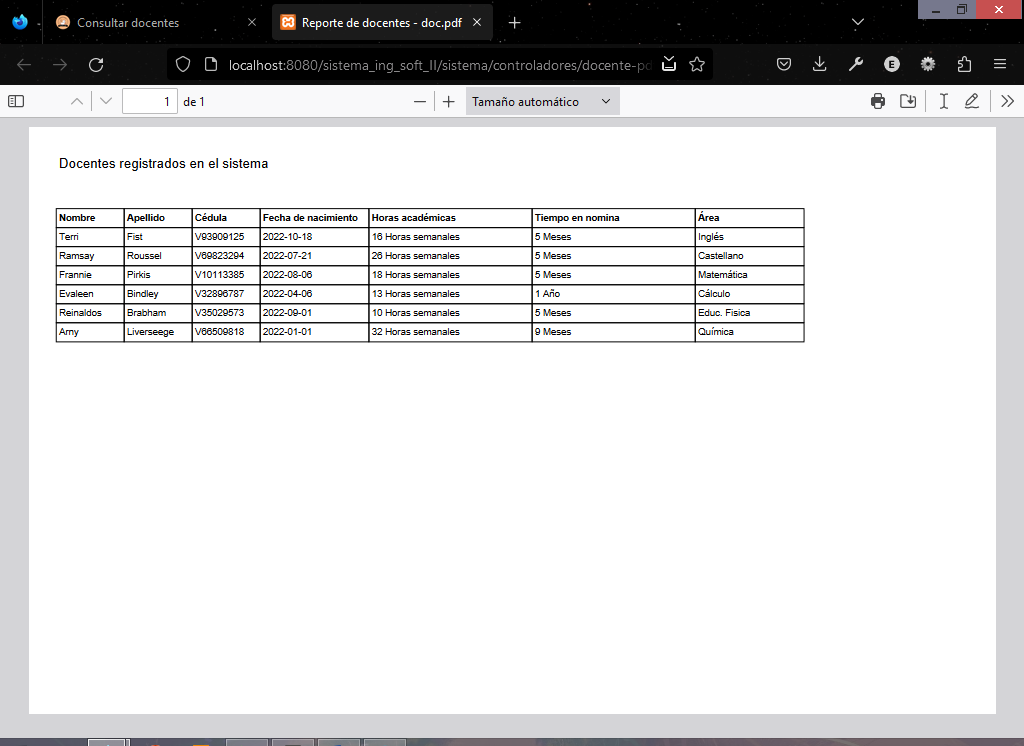


Se establece la tipografía como Arial de 10 puntos en negrita, luego se imprimen celdas de tamaño fijo y con borde y un salto de línea.

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.

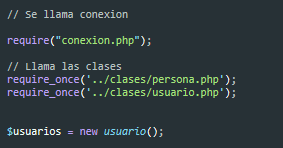
Finalmente se imprime el documento completo para que el usuario lo pueda visualizar o descargar (dependiendo de la configuración del navegador).



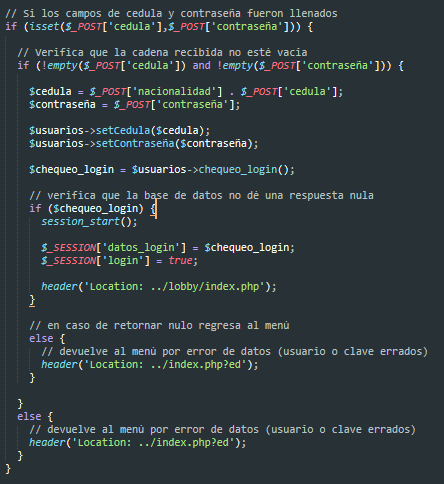


### login.php

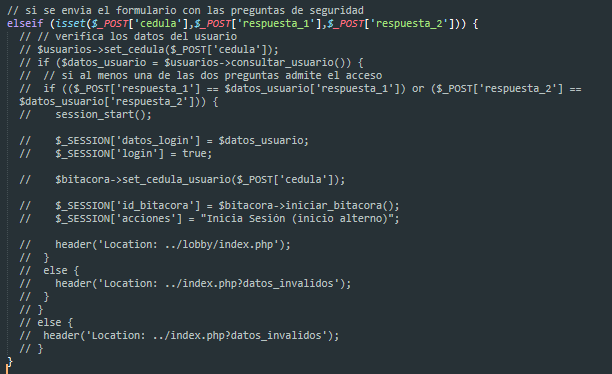
Verifica los datos de ingreso del usuario (Cédula y contraseña), en caso de ser correcto, inicia una sesión en el sistema y concede el acceso al usuario.



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



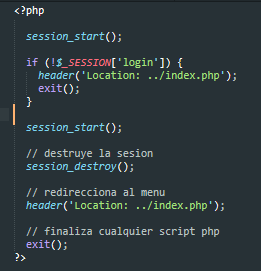
Se verifica que se haya enviado tanto la cédula como la contraseña del usuario y que estos no estén vacíos, en caso de no haberse enviado o estén vacíos, envía al usuario a la pantalla de inicio de sesión, si fueron enviados concatena la nacionalidad con el número de cédula del usuario, y se asigna su 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.



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

### logout.php

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



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

## 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.

### Login

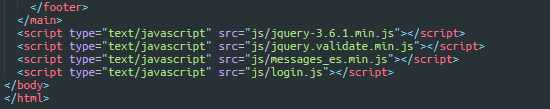
Se encuentra alojada en index.php dentro de la carpeta sistema.



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.



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



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.



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:



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

### 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:

* 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.

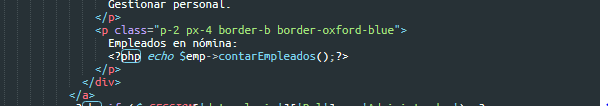
Fuera del maquetado, esta vista cuenta con los siguientes aspectos de programación.



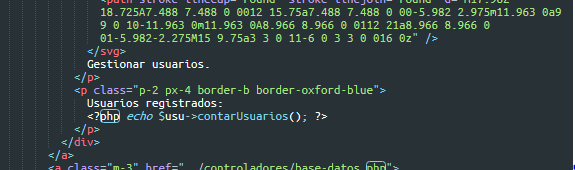
En un comienzo, se retoma la sesion iniciada en login.php, verifica si el usuario ha iniciado sesion (para evitar ingresar desde la url sin iniciar sesion). Requiere conexión.php, empleado.php y usuario.php e instancia la conexión a base de datos para luego comprobar el estado de la base de datos, luego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



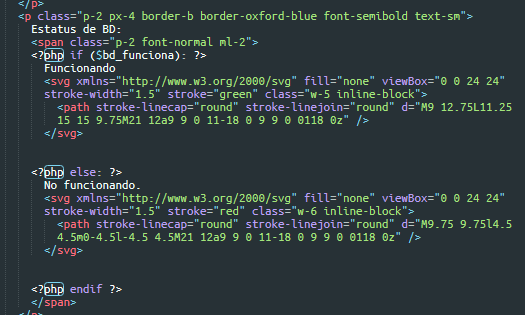
Muestra el nombre y apellido del usuario en un mensaje de bienvenida.



Muestra el número de empleados registrados.



Muestra el número de usuarios registrados.



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

### Consultar.

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

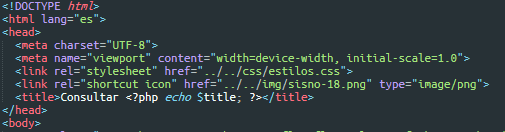
Este módulo se encuentra en la carpeta consulta y cuenta con cinco variantes y un submódulo que permite hacer una consulta individual y especifica del empleado.

Nota: todas la 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

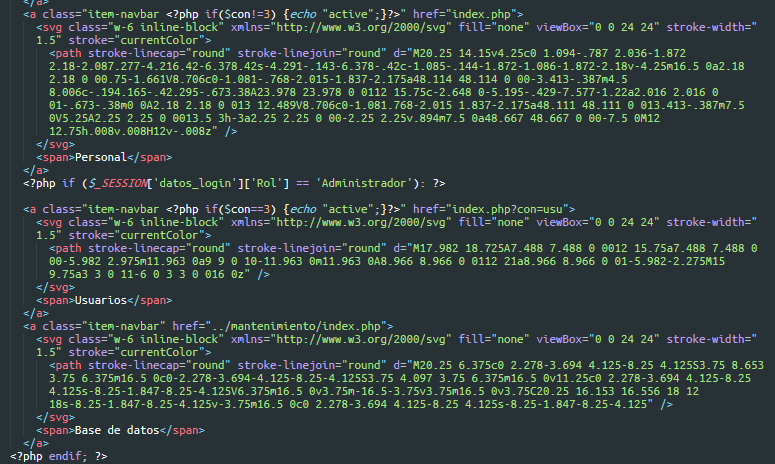
A nivel de programación este módulo hace las siguientes acciones:



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.



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



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 resaltará la pestaña usuarios.



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

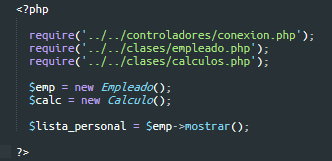
#### Consulta de personal

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

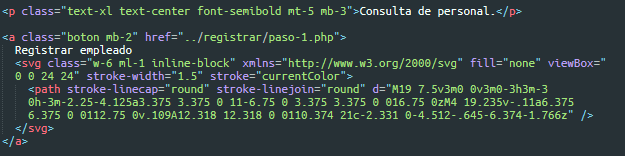


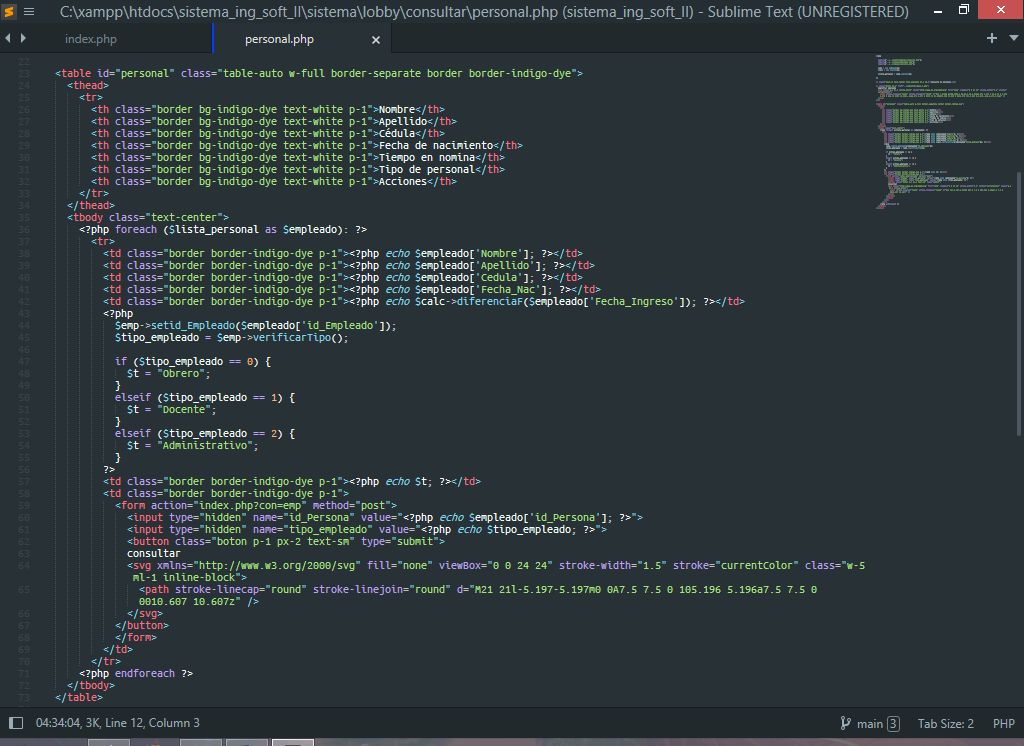
Muestra los datos de generales de los empleados registrados en sistema, teniendo en cuenta nombres, apellidos, cédula, fecha de nacimiento, cuánto tiempo lleva en nómina de una manera aproximada y que tipo de personal es.

A nivel de programación, hace lo siguiente:



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.





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 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 evaluara 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.

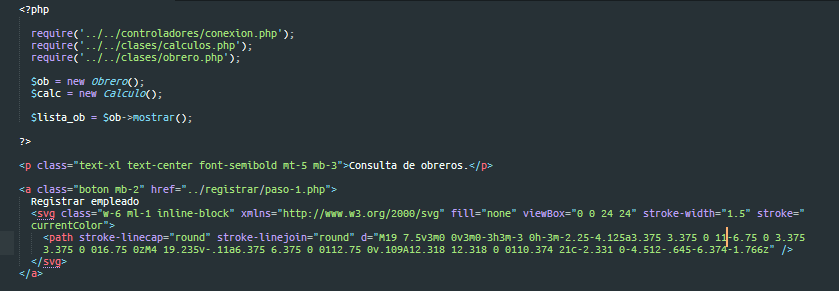
#### Consulta de obreros

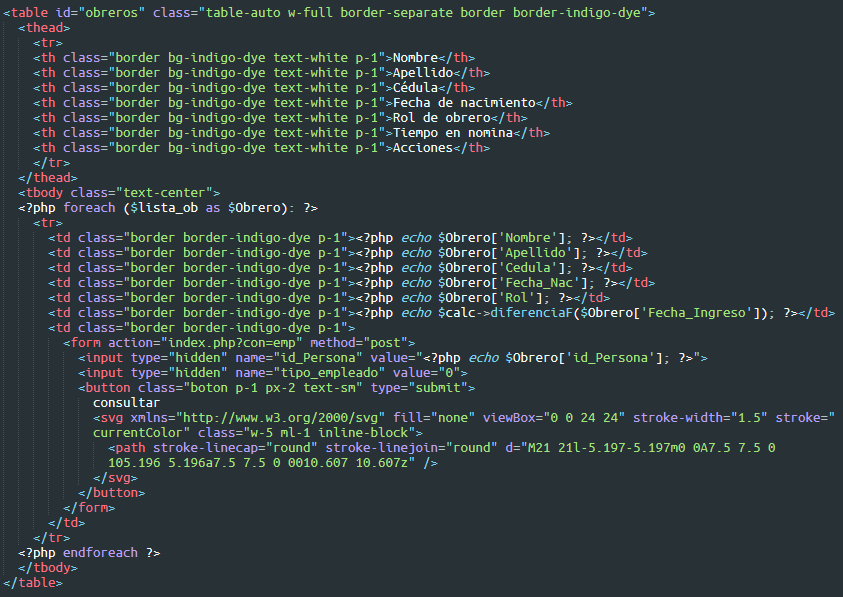
En el caso de que $con sea igual a 0, se requiere el archivo obreros.php como un agregado a index.php



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.

A nivel de programación, hace lo siguiente:





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.

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.

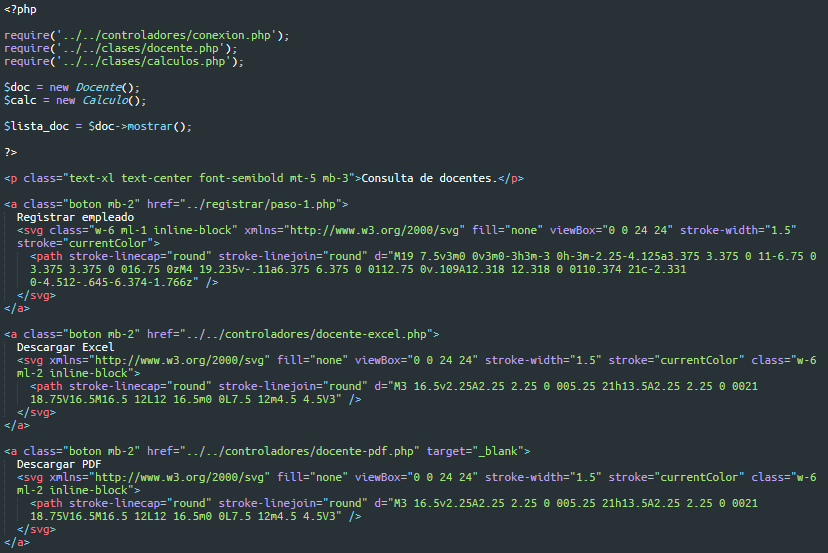
#### Consulta de docentes

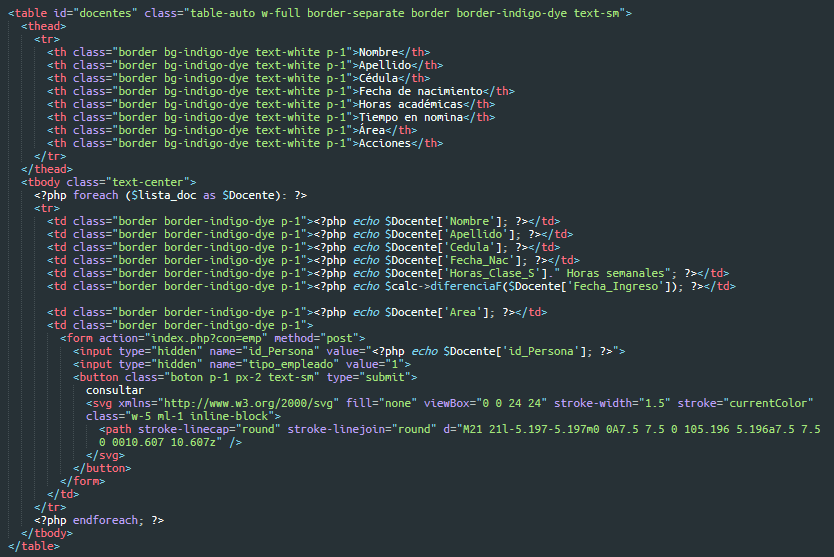
En el caso de que $con sea igual a 1, se requiere el archivo docentes.php como un agregado a index.php



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:





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.

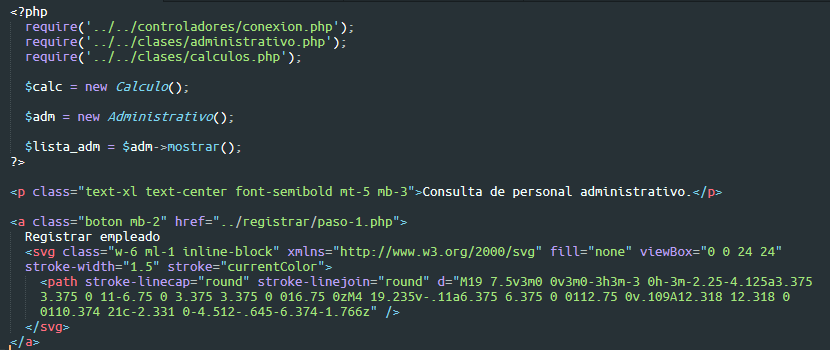
En la columna de acción se encuentra un formulario con un botón que envía el identificador y el tipo de empleado al mismo index.php pasando una variable GET que especifica que se va a consultar ese empleado.

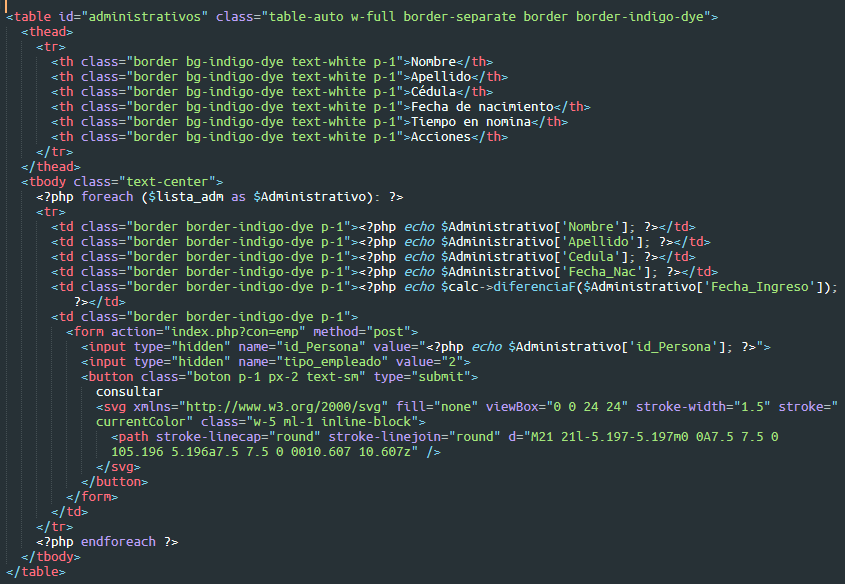
#### Consulta de personal administrativo



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:





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.

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.

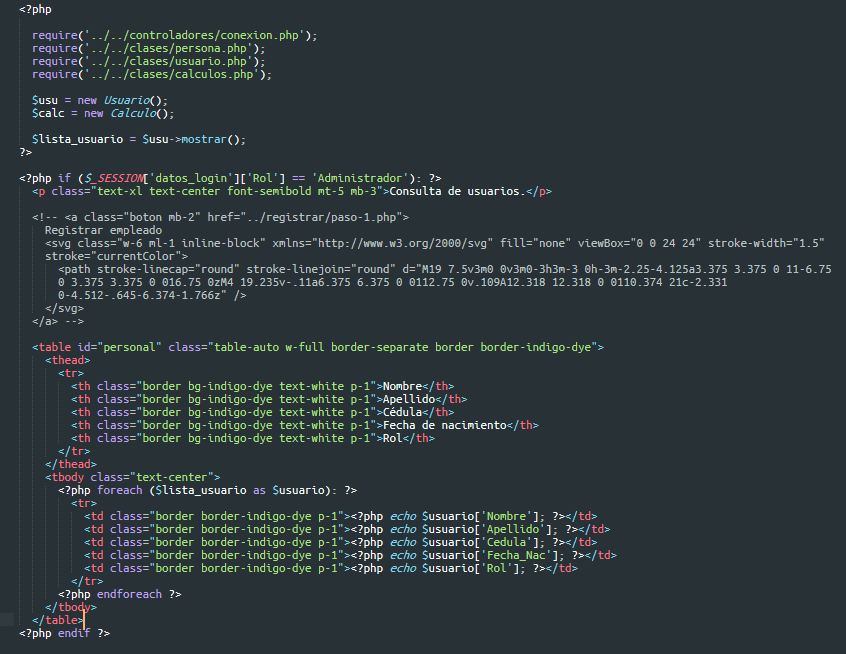
En la columna de acción se encuentra un formulario con un botón que envía el identificador y el tipo de empleado al mismo index.php pasando una variable GET que especifica que se va a consultar ese empleado.

#### Consulta de usuarios



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.

A nivel de programación, hace lo siguiente:

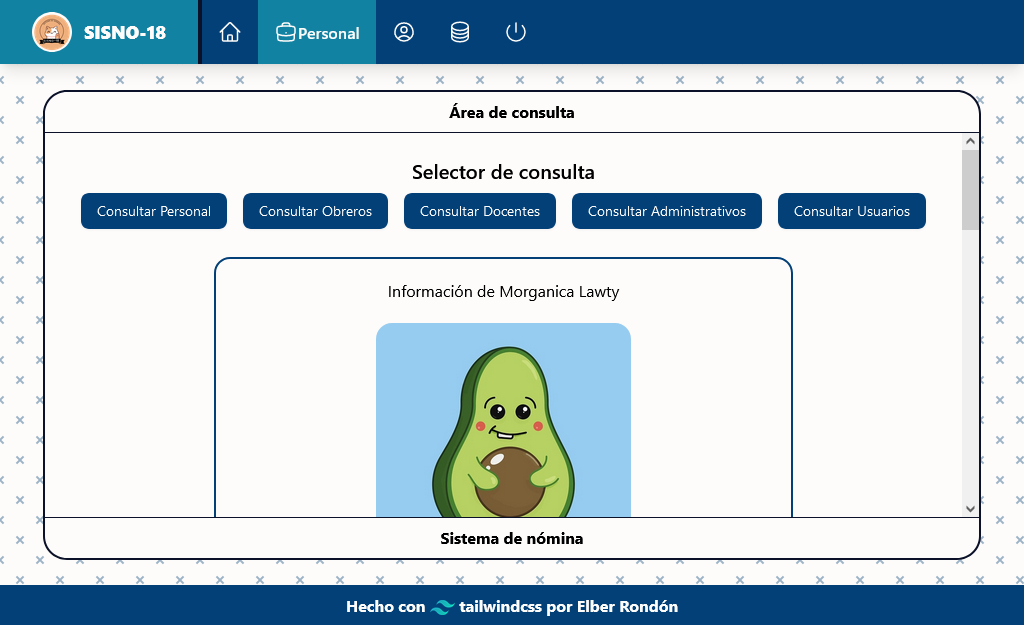


Inicia requiriendo el controlador de conexión a base de datos y los archivos de usuario y cálculos, instancia estas clases y usa el método mostrar de la clase Usuario para almacenar la lista de usuarios registrados.

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.

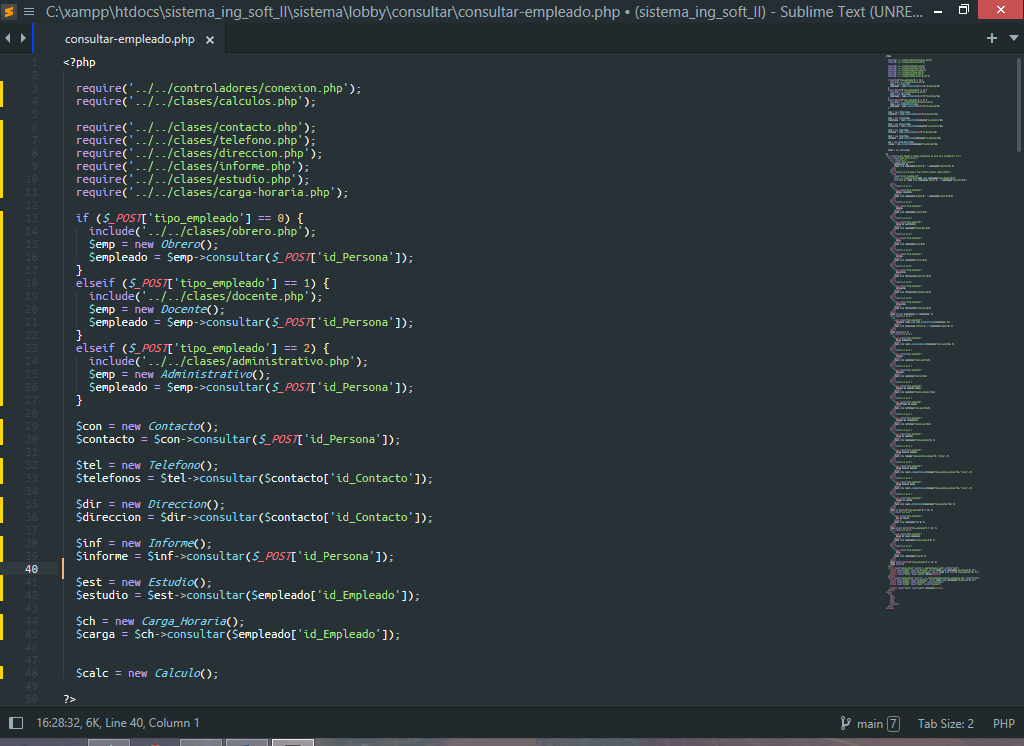
Cabe mencionar que la tabla solo se mostrará si el usuario que solicita la consulta es un administrador. De lo contrario, no lo hará.

#### Consultar empleado



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

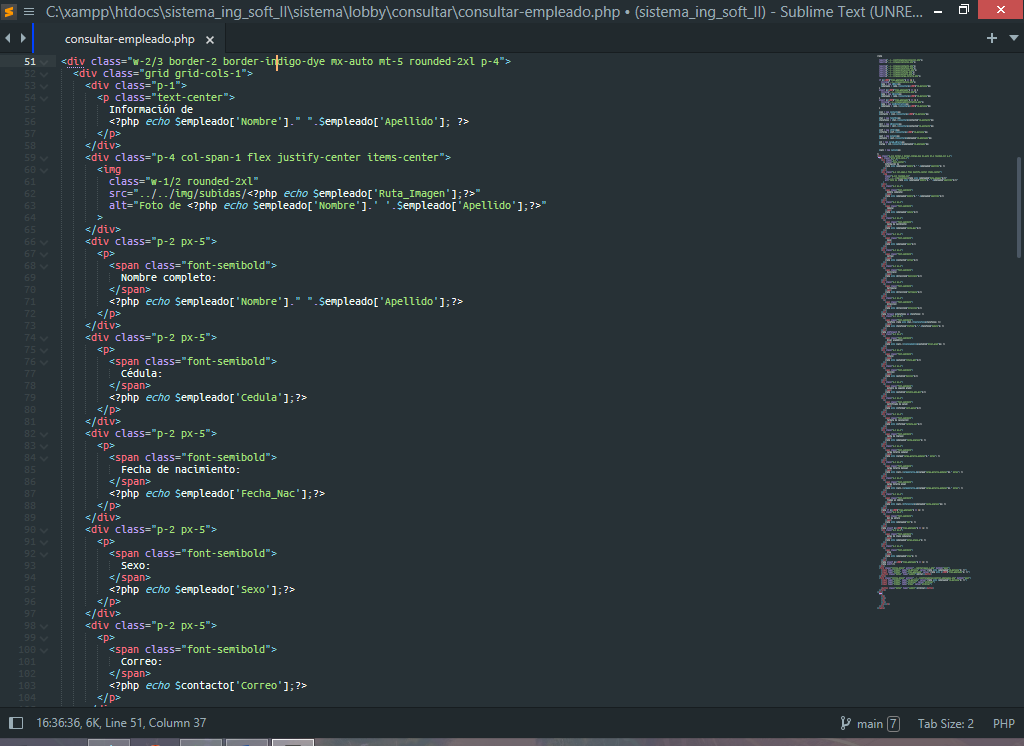
A nivel de programación hace lo siguiente:



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.

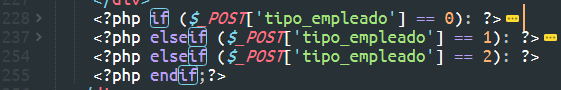
Hará consultas de igual manera con el resto de clases hasta traer todos los datos correspondientes al empleado y finalmente instancia la clase Calculo.



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 | ✓ | ✓ | ✓ |
| 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



Esta misma separacion se ve en las siguientes condiciones.

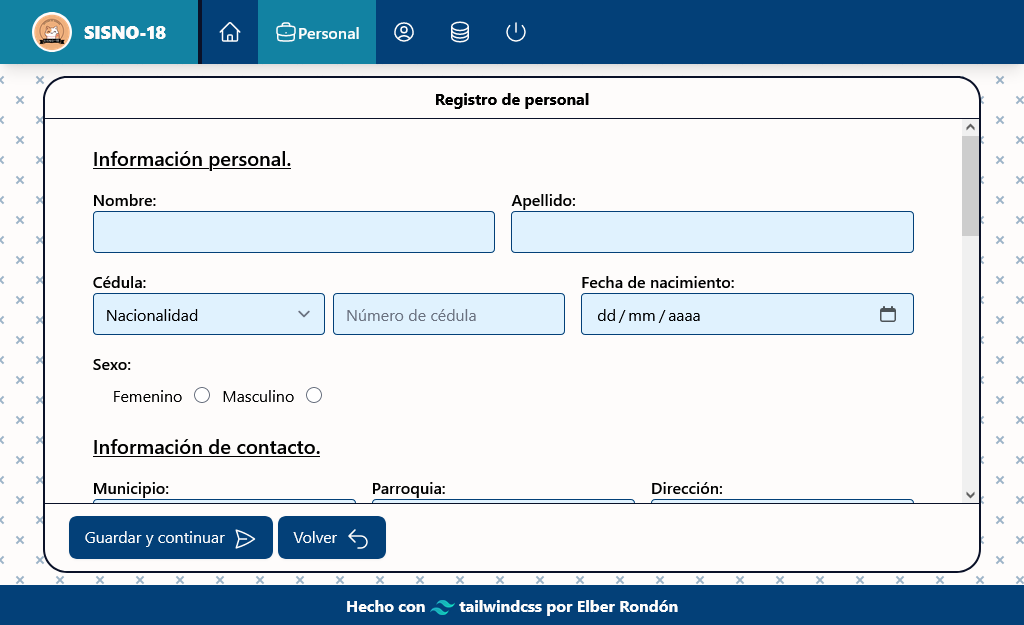


Por último, tambien se muestran dos botónes en pantalla que permiten editar o eliminar al empleado de la base de datos.

### Registrar

Consta de un formulario de dos partes.

#### Paso 1

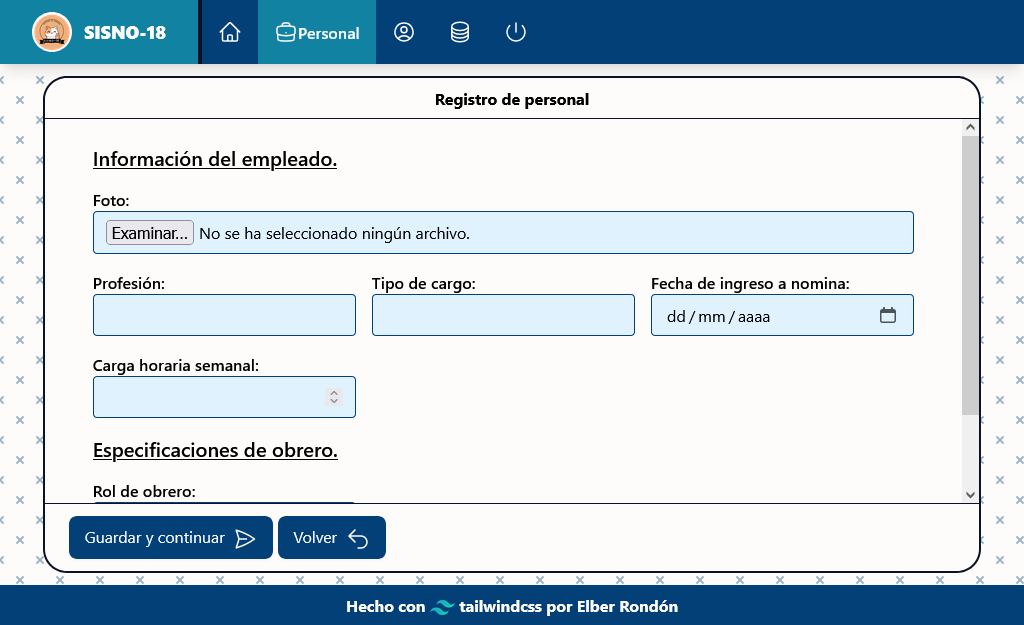


Vista principal del paso 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |
| 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



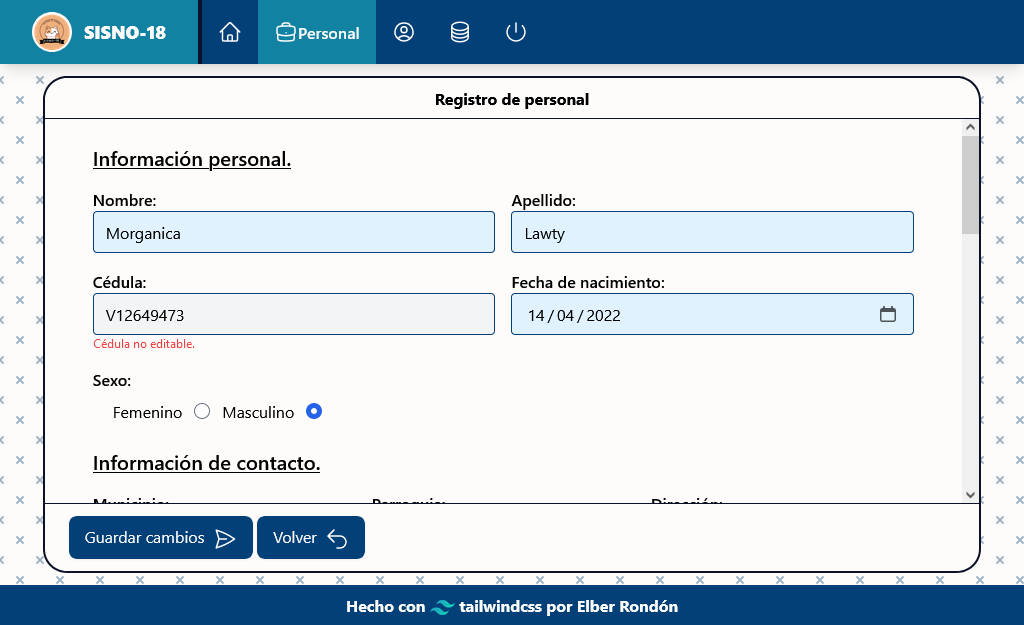
Vista del segundo paso del formulario de registro.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| mencion | hidden |  | Recibido del paso anterior |
| E\_2do\_nivel | hidden |  | Recibido del paso anterior |
| 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 |  |

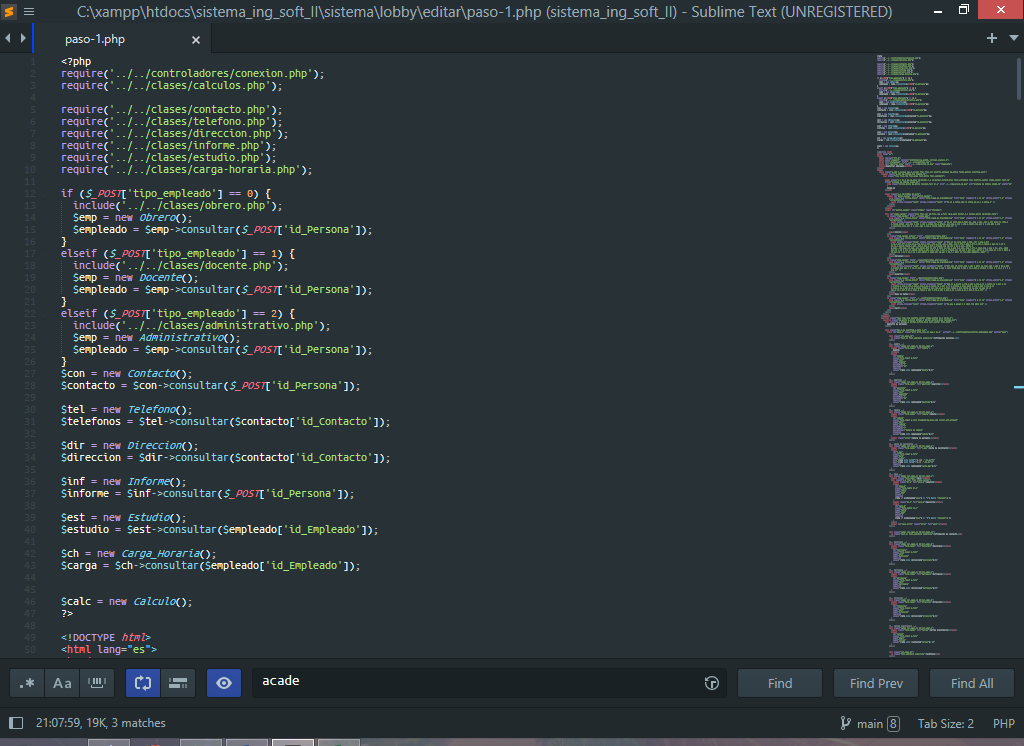
Una vez que el usuario ingrese los datos del nuevo empleado, estos seran enviados al controlador contro-empleados.php para su ingreso en la base de datos.

### Editar

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



Vista general del formulario de edición.



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 |  |  | |
| 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.