

Carrera: Ingeniería en las Tecnologías de la Información

Asesor: Heriberto Rene Saldaña

Alumnos:

Yuritzi Guadalupe Cañamar Velázquez

Leonel Alberto Leija Pérez

**“Manual Técnico”**

Contenido

[**Introducción** 1](#_Toc61232525)

[**Requerimientos técnicos** 2](#_Toc61232526)

[**Herramientas utilizadas** 2](#_Toc61232527)

[**PHP** 2](#_Toc61232528)

[**XAMPP** 3](#_Toc61232529)

[**phpMyAdmin** 3](#_Toc61232530)

[**Sublime Text** 4](#_Toc61232531)

[**Instalar XAMPP** 4](#_Toc61232532)

[**Sistema** 7](#_Toc61232533)

[**Explicación del código y las interfaces** 8](#_Toc61232534)

[**index.php** 8](#_Toc61232535)

[**.htaccess** 10](#_Toc61232536)

[**Controladores** 10](#_Toc61232537)

[**Controlador ctrlguardarInv.php** 11](#_Toc61232538)

# **Introducción**

El propósito de este documento será de explicar el código y las interfaces del proyecto de estadías que se realizó durante el periodo del mismo.

Puede que algún código o algunas interfaces de omitan por petición de la empresa dado que puede tener información clasificada de la misma.

# **Requerimientos técnicos**

Para hacer uso de la página web que se desarrolló para la empresa, se necesitara uno de los siguientes dos requerimientos

**Requerimiento 1:**

Descargar algún servidor local para usarlo de personal (PC).

Descargar e instalar un gestor de bases de datos (ej.: phpmyadmin).

**Requerimiento 2:**

Buscar un host para subir la página web.

Tener la base de datos lista y configurada para subirla con junto a la página web.

# **Herramientas utilizadas**

## **PHP**



Img 1.0 Logo de PHP

Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo" (en este caso, mostrar "¡Hola, soy un script de PHP!). El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

## **XAMPP**



Img 1.1 Logo de 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.

## **phpMyAdmin**



Img 1.3 Logo de phpMyAdmin

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador web. Para entender mejor, es un gestor de bases de datos.

## **Sublime Text**



Img 1.4 Logo de Sublime Text

Sublime Text es un editor de texto y editor de código fuente. Está escrito en C++ y Python para los plugins.​ Desarrollado originalmente como una extensión de Vim, con el tiempo fue creando una identidad propia. Aún conserva un modo de edición tipo vi llamado Vintage mode. Con este editor se puede utilizar diferentes tipos de lenguajes, en este caso, se utilizó para escribir PHP.

# **Instalar XAMPP**

Paso 1: Descargar XAMPP.

Paso 2: Correr el ejecutable .exe.

Paso 3: Esperar que inicie el asistente de instalación.



Img 2.0 Asistente de XAMPP

Paso 4: Seleccionar que compontes instalar.



Img 2.1 XAMPP componentes

Paso 5: Seleccionar en donde se instalará XAMPP



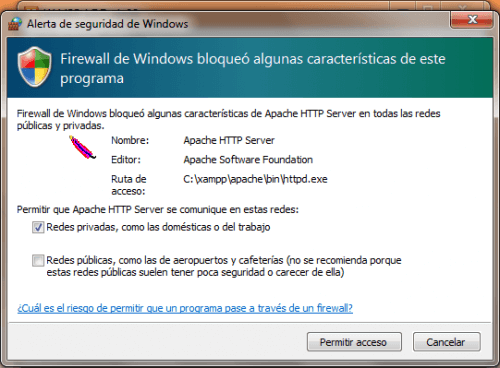
Img 2.2 XAMPP localización

Paso 6: Esperar a que se instale XAMPP



Img 2.3 XAMPP instalación

Paso 7: Dar los permisos necesarios:



Img 2.4 XAMPP permisos

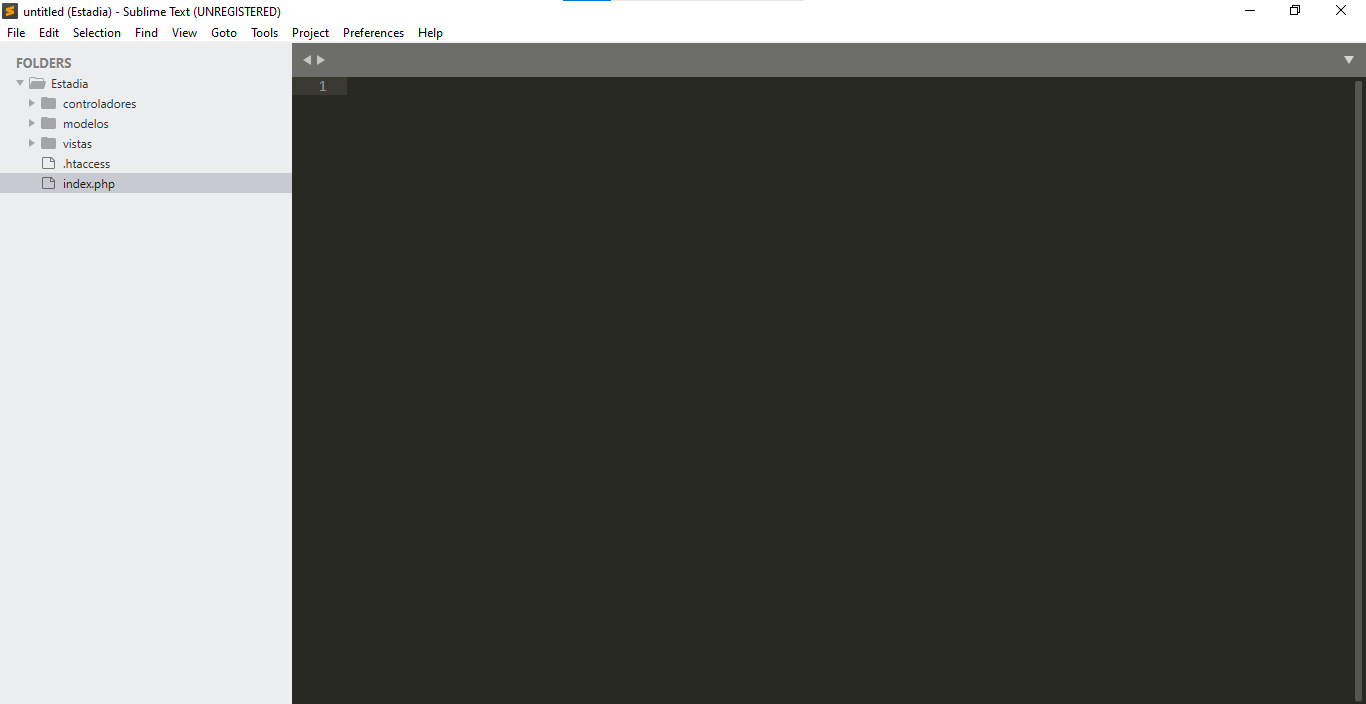
Paso 8: Confirmar la instalación:



Img 2.5 confirmación de la instalación

# **Sistema**

El sistema que se desarrollo para la empresa, se contemplo que todo estuviera de forma más acomodada y de forma un poco más profesional. Por lo cual, se opto por utilizar el modelo MVC parra tener todo organizado y con un poco más de clasificación y seguridad para el mismo.



Img 3.0 Sistema MVC

## **Explicación del código y las interfaces**

## **index.php**

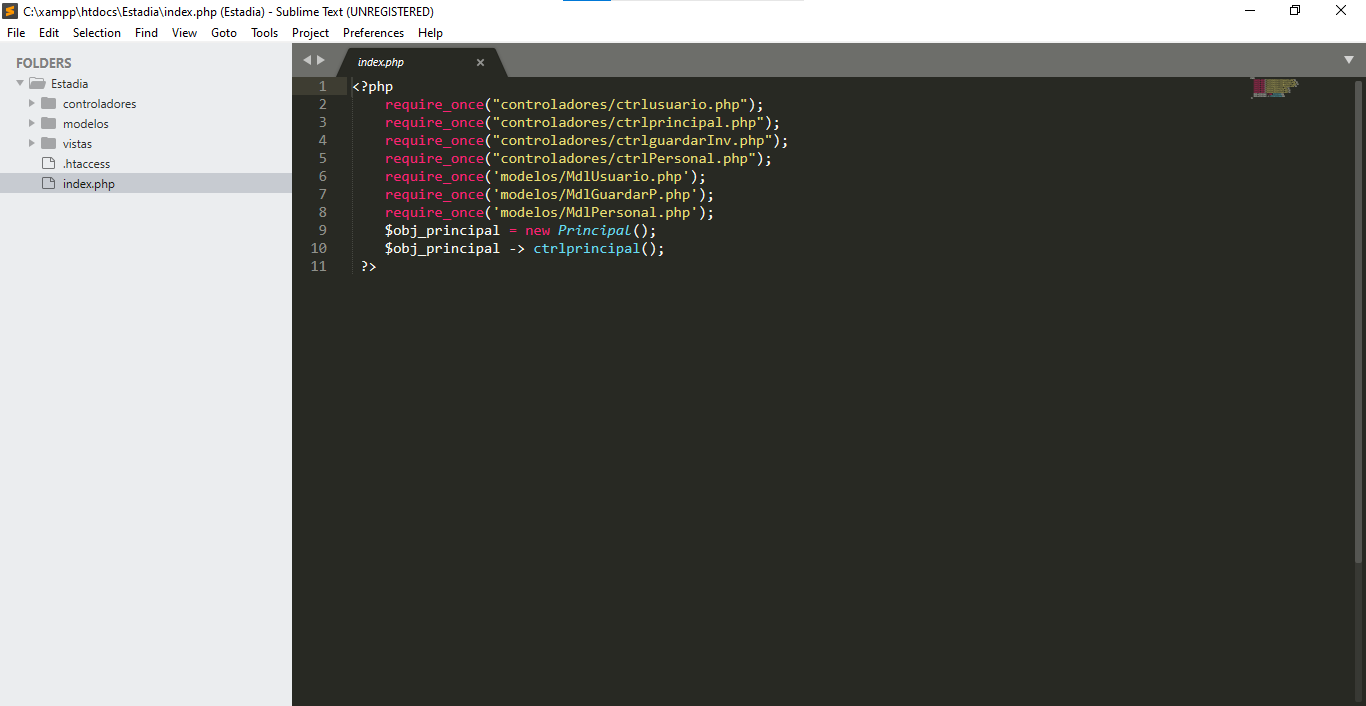
Este archivo nos permitirá que al momento de entrar al proyecto nos mande directamente al login, además de que contiene todos los demás archivos que se utilizarán en otros archivos .php que los necesitan o llamarían con un include o requiere\_once.

Se hizo de esta manera por dos motivos:

1. Por seguridad.
2. Para organización.

Seguridad: Si alguien tiene los conocimientos para entrar y ver el código fuente, de esta manera se le complicara saber que archivo llama o necesita otro archivo php.

Organización: Para evitar que en cada archivo se llame un archivo cada vez que se necesite, solo ponemos el archivo en el index y con eso bastara para todas aquellas páginas que necesiten un mismo archivo.



Img 3.1 código dentro del index,php

Ahora, ya se explicó por qué se tiene los requiere\_once en el index, se explicará que es lo que hace el código php.

Código:

// Forma de utilizar el lenguaje de php es abrir y cerrar <php >? para poder escribir código dentro. De otra manera, no se podría utilizar el código dentro y las líneas de código no funcionarían.

<php

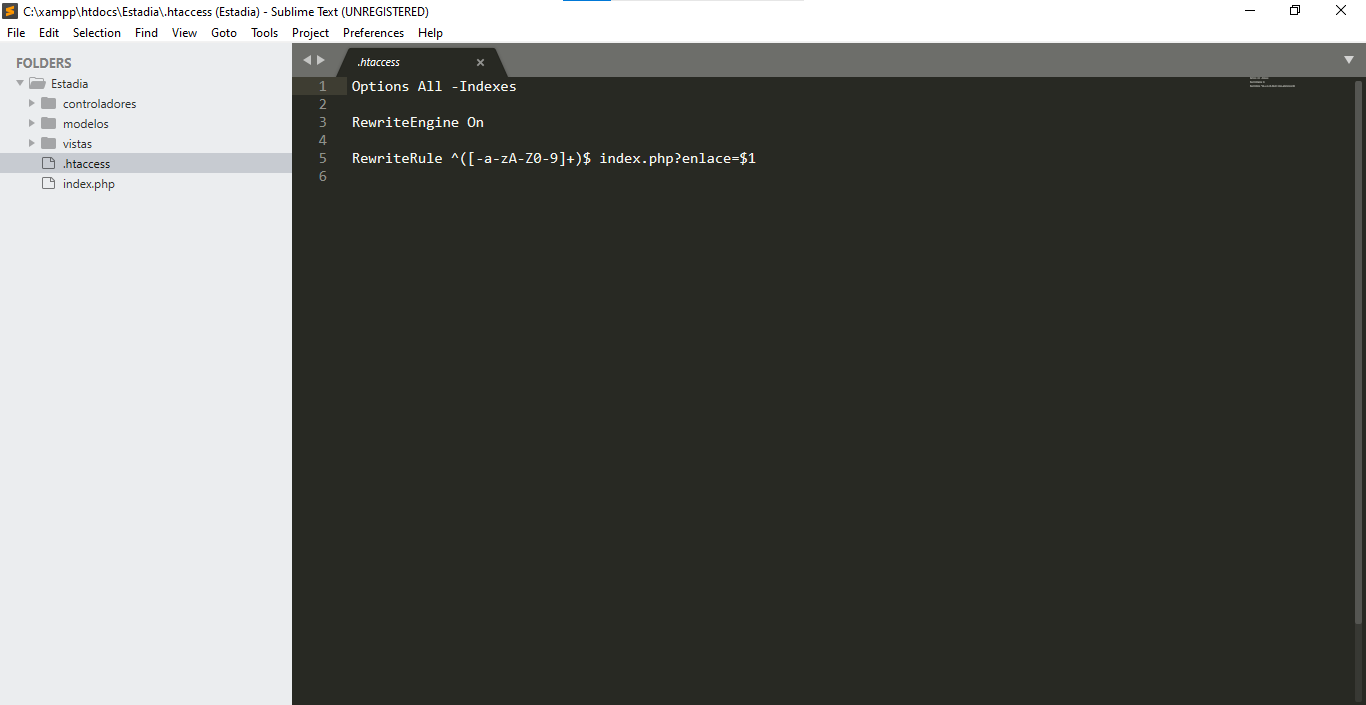
$obj\_principal = new principal(); // Permite que se cree una instancia de una clase de un controlador que tenga el nombre que estamos instanciando para posteriormente poder utilizar los métodos de esa clase.

$obj\_principal -> ctrlprincipal(); // Hacemos uso del método de la clase principal para poder acceder al login y para después meter datos.

?>

Antes de poder explicar lo que sería los controladores a donde nos manda el index, se tiene que explicar el archivo .htaccess.

## **.htaccess**



Img 3.2 .htaccess

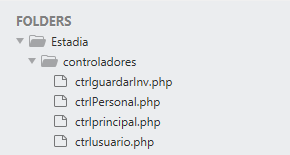
Con .htaccess, nos aseguramos que solo se pueda introducir números y letras, y excluir los caracteres especiales.

Estoy al momento de tener que seleccionar alguna pagina del sistema, en vez de que vaya a una que no existe y salga error en el navegador, esta nos llevara a otra conocida como pagina 404 en donde muestra que tal página no existe.

Esta misma trabaja con las interfaces que se verán más adelante.

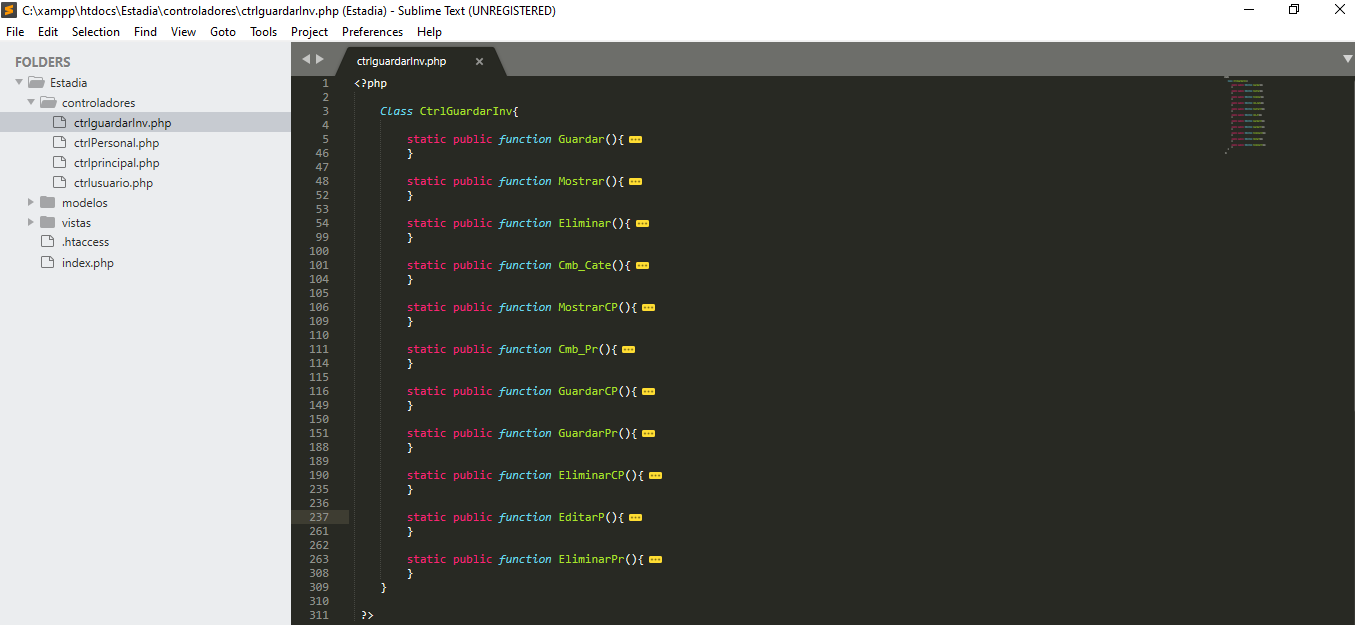
## **Controladores**

Lo que sigue es explicar las carpetas y los archivos .php, sin embargo, tener que explicarlo de forma lógica tomaría mucho tiempo y tal vez se perdería el hilo de la explicación. Por ello, se explicará todo de forma ordenada, de arriba hacia abajo. Por ello se explicará primero los controladores.



Img 4.0 Controladores

### **Controlador ctrlguardarInv.php**



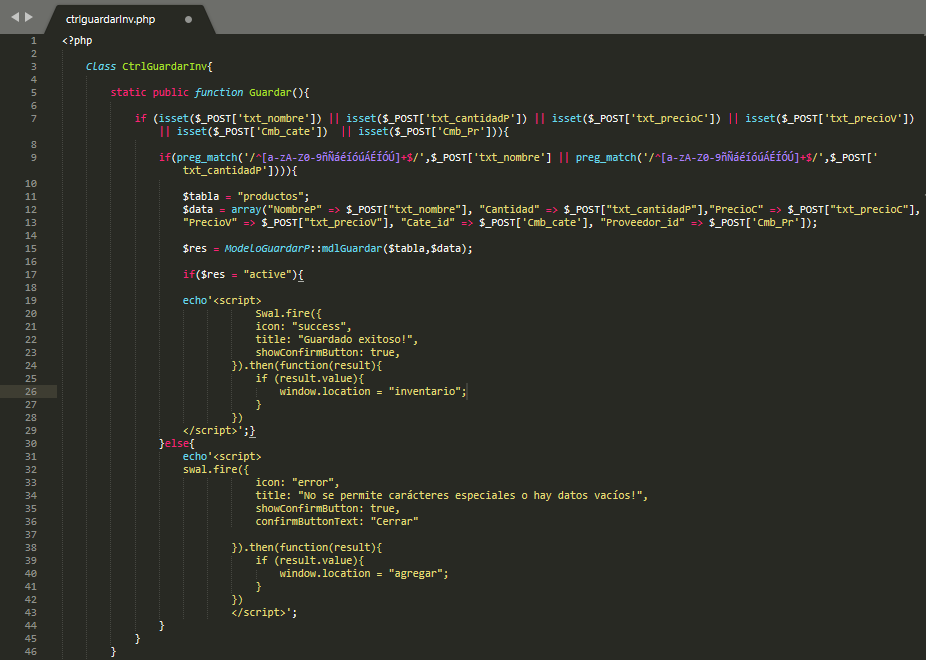
Img 4.1 controlador ctrlguardarinv.php

A continuación, se muestra los controladores que permitirán que se pasen los datos entre las interfaces y podamos validarlos para entrar a los modelos.

Por la cantidad de controladores que hay y lo extenso que es el código en cada método, se explicara solo los métodos de una interface y los demás solo se mostraran sin explicar.

Hay que decir que los demás métodos son similares a los que san a explicar, por ello no se pierde tanto la idea o el razonamiento de los mismos.

**Método Guardar:**

****

**Img 4.2 cotrlguardarInv método guardar**

Para poder comentar el código pero que a la vez se pueda entender con el acomodo que debe tener, se pondrá una imagen del código original y a la vez el código acomodado para poder comentarlo.

static public function Guardar(){ //creación del método

//condicional en donde al momento que el post reciba los datos, este confirme que no vengan vacíos.

if (isset($\_POST['txt\_nombre']) || isset($\_POST['txt\_cantidadP']) || isset($\_POST['txt\_precioC']) || isset($\_POST['txt\_precioV']) || isset($\_POST['Cmb\_cate']) || isset($\_POST['Cmb\_Pr'])){

//Segundo condicional en donde se comprueba que los datos dentro de la condicional no contengan caracteres especiales.

if(preg\_match('/^[a-zA-Z0-9ñÑáéíóúÁÉÍÓÚ]+$/',$\_POST['txt\_nombre'] || preg\_match('/^[a-zA-Z0-9ñÑáéíóúÁÉÍÓÚ]+$/',$\_POST['txt\_cantidadP']))){

$tabla = "productos";//Asignación de valor con el nombre de la tabla para mandar al modelo.

$data = array("NombreP" => $\_POST["txt\_nombre"], "Cantidad" => $\_POST["txt\_cantidadP"],"PrecioC" => $\_POST["txt\_precioC"],

"PrecioV" => $\_POST["txt\_precioV"], "Cate\_id" => $\_POST['Cmb\_cate'], "Proveedor\_id" => $\_POST['Cmb\_Pr']); //En una variable de tipo array, asignamos todos los valores con una asignación para que de forma segura solo se mande la variable array y no se vea los valores que se estan trayendo con el POST

$res = ModeloGuardarP::mdlGuardar($tabla,$data);//Mandamos el nombre de la tabla y el arrray hacia el modelo correspondiente para guardar los valores en la base de datos.

if($res = "active"){ // Condición en donde dependiendo de la respuesta que nos mande el modelo, se mostrara un mensaje

echo'<script>

Swal.fire({//Sweet alert, mensaje bonitos para el sistema

icon: "success",

title: "Guardado exitoso!",

showConfirmButton: true,

}).then(function(result){

if (result.value){

window.location = "inventario";

}

})

</script>';}

}else{

echo'<script>

swal.fire({

icon: "error",

title: "No se permite carácteres especiales o hay datos vacíos!",

showConfirmButton: true,

confirmButtonText: "Cerrar"

}).then(function(result){

if (result.value){

window.location = "agregar";

}

})

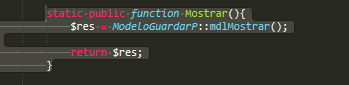
</script>';

}

}

}

**Método Mostrar:**

****

**Img 4.3 ctrlguardarInv método Mostrar**

Este método, como en los demás métodos de otras interfaces, es el más fácil de comprender, dado que solo está pidiendo que se muestre los datos en una consulta de una tabla en específico, así que, por ello, no se mandan datos desde la interface.

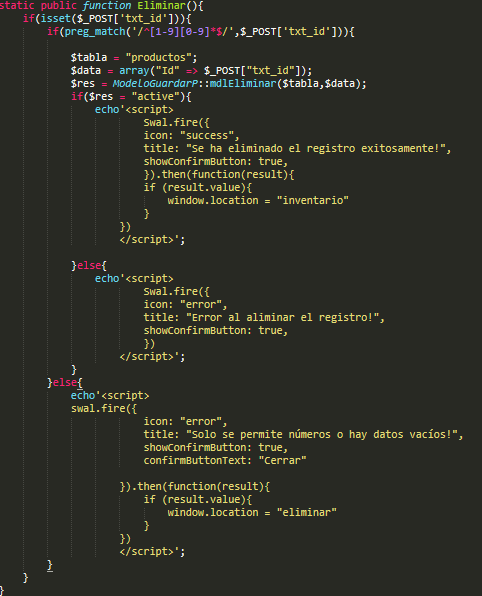
static public function Mostrar(){// creación del método

$res = ModeloGuardarP::mdlMostrar();//Lo que se está utilizando aquí es que, en vez de crear un objeto para utilizar la función, solo utilizamos la resolución de alcance para llamar al método con una variable.

return $res; //Se regresa la variable con la respuesta o el valor que haya arrojado la función.

}

**Método Eliminar**

****

**Img 4.4 método Eliminar**

Como el nombre lo dice, el método eliminar toma solo el id del registro para poder eliminarlo. Se puede apreciar que hay varios else, uno es la confirmación de que se ha eliminado el registro, otro que habido un error y el ultimo que nos dice que hay caracteres especiales o alfabéticos y por ello no se puede eliminar.

static public function Eliminar(){//Inicio del método

if(isset($\_POST['txt\_id'])){//if en donde se asegura que el id no venga vacio

if(preg\_match('/^[1-9][0-9]\*$/',$\_POST['txt\_id'])){//evaluación del método if

$tabla = "productos";//Tabla donde se encuentra el registro

$data = array("Id" => $\_POST["txt\_id"]);//Asignamos el valor del id a una variable

$res = ModeloGuardarP::mdlEliminar($tabla,$data);//Método en donde se manda el id para eliminar

if($res = "active"){//Mensaje de confirmación de eliminado

echo'<script>

Swal.fire({

icon: "success",

title: "Se ha eliminado el registro exitosamente!",

showConfirmButton: true,

}).then(function(result){

if (result.value){

window.location = "inventario"

}

})

</script>';

}else{

echo'<script>

Swal.fire({//Mensaje de que no se ha podido eliminar el registro

icon: "error",

title: "Error al aliminar el registro!",

showConfirmButton: true,

})

</script>';

}

}else{

echo'<script>

swal.fire({//Mensaje de que hay caracteres especiales o alfabéticos

icon: "error",

title: "Solo se permite números o hay datos vacíos!",

showConfirmButton: true,

confirmButtonText: "Cerrar"

}).then(function(result){

if (result.value){

window.location = "eliminar"

}

})

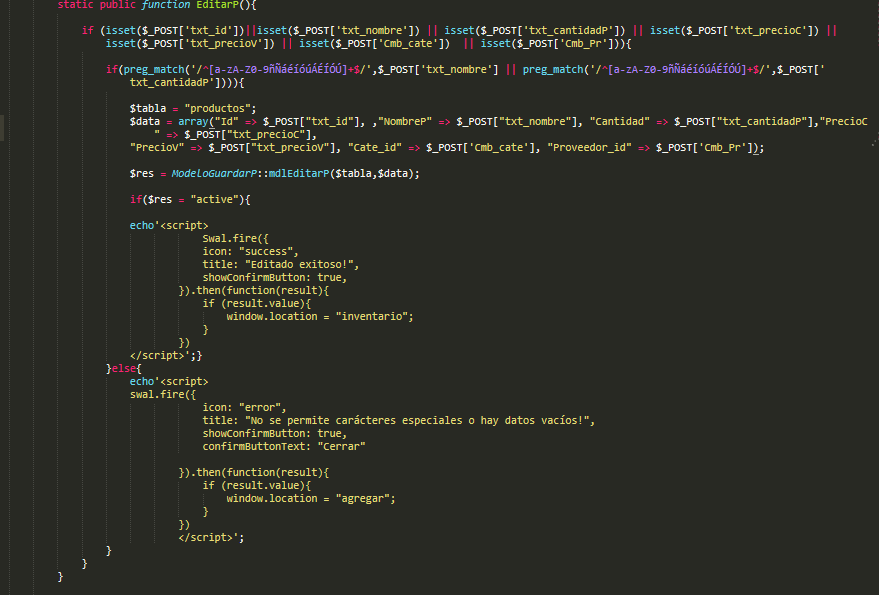
</script>';

}

}

}

**Método editar**

****

**Img4.5 Método editar**

Se puede observar que el método editar o actualizar es similar al método de guardar, la única diferencia aquí es que se trae de la interface el id con el cual podremos buscar el registro y actualizarlo. El código en si es el mismo, solo con el id de más así que se puede ver la explicación del método guardar para entender el siguiente código.

static public function EditarP(){

if (isset($\_POST['txt\_id'])||isset($\_POST['txt\_nombre']) || isset($\_POST['txt\_cantidadP']) || isset($\_POST['txt\_precioC']) || isset($\_POST['txt\_precioV']) || isset($\_POST['Cmb\_cate']) || isset($\_POST['Cmb\_Pr'])){

if(preg\_match('/^[a-zA-Z0-9ñÑáéíóúÁÉÍÓÚ]+$/',$\_POST['txt\_nombre'] || preg\_match('/^[a-zA-Z0-9ñÑáéíóúÁÉÍÓÚ]+$/',$\_POST['txt\_cantidadP']))){

$tabla = "productos";

$data = array("Id" => $\_POST["txt\_id"], ,"NombreP" => $\_POST["txt\_nombre"], "Cantidad" => $\_POST["txt\_cantidadP"],"PrecioC" => $\_POST["txt\_precioC"],

"PrecioV" => $\_POST["txt\_precioV"], "Cate\_id" => $\_POST['Cmb\_cate'], "Proveedor\_id" => $\_POST['Cmb\_Pr']);

$res = ModeloGuardarP::mdlEditarP($tabla,$data);

if($res = "active"){

echo'<script>

Swal.fire({

icon: "success",

title: "Editado exitoso!",

showConfirmButton: true,

}).then(function(result){

if (result.value){

window.location = "inventario";

}

})

</script>';}

}else{

echo'<script>

swal.fire({

icon: "error",

title: "No se permite carácteres especiales o hay datos vacíos!",

showConfirmButton: true,

confirmButtonText: "Cerrar"

}).then(function(result){

if (result.value){

window.location = "agregar";

}

})

</script>';

}

}

}

Habiendo explicado los métodos principales de la interface de productos, se ira agregando los demás métodos que no tendrán explicación porque en si son lo mismo, pero con diferente nombre en el método.

**Método de Cmb\_Cate**



Img 4.6 método combobox

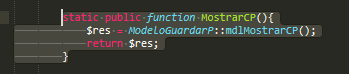
static public function Cmb\_Cate(){

$res = ModeloGuardarP::mdlCmb\_Cate();

return $res;

}

**Método MostrarCP**



Img 4.6 Método MostarCP

static public function MostrarCP(){

$res = ModeloGuardarP::mdlMostrarCP();

return $res;

}

**Método Cmb\_Pr**

****

Img 4.7 Método Cmb\_Pr

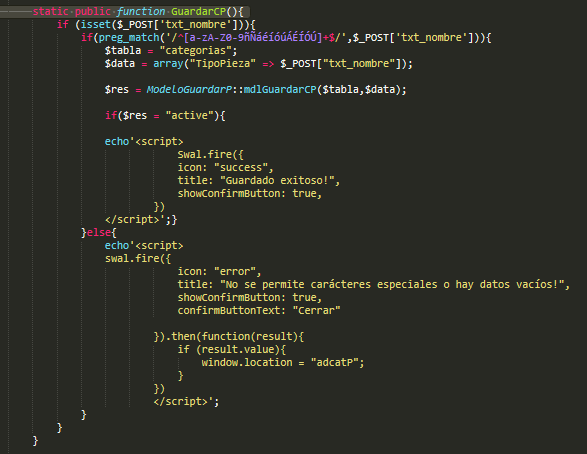
static public function Cmb\_Pr(){

$res = ModeloGuardarP::mdlCmb\_Pr();

return $res;

}

**Método GuardarCP**



Img 4.8 método GuardarCP

static public function GuardarCP(){

if (isset($\_POST['txt\_nombre'])){

if(preg\_match('/^[a-zA-Z0-9ñÑáéíóúÁÉÍÓÚ]+$/',$\_POST['txt\_nombre'])){

$tabla = "categorias";

$data = array("TipoPieza" => $\_POST["txt\_nombre"]);

$res = ModeloGuardarP::mdlGuardarCP($tabla,$data);

if($res = "active"){

echo'<script>

Swal.fire({

icon: "success",

title: "Guardado exitoso!",

showConfirmButton: true,

})

</script>';}

}else{

echo'<script>

swal.fire({

icon: "error",

title: "No se permite carácteres especiales o hay datos vacíos!",

showConfirmButton: true,

confirmButtonText: "Cerrar"

}).then(function(result){

if (result.value){

window.location = "adcatP";

}

})

</script>';

}

}

**Método GuardarPr**

****

**Img 4.9 método GuardarPr**

static public function GuardarPr(){

if (isset($\_POST['txt\_nombre'])){

if(preg\_match('/^[a-zA-Z0-9ñÑáéíóúÁÉÍÓÚ]+$/',$\_POST['txt\_nombre'])){

$tabla = "proveedor";

$data = array("Proveedor" => $\_POST["txt\_nombre"]);

$res = ModeloGuardarP::mdlGuardarPr($tabla,$data);

if($res = "active"){

echo'<script>

Swal.fire({

icon: "success",

title: "Guardado exitoso!",

showConfirmButton: true,

}).then(function(result){

if (result.value){

window.location = "adcatPr";

}

})

</script>';}

}else{

echo'<script>

swal.fire({

icon: "error",

title: "No se permite carácteres especiales o hay datos vacíos!",

showConfirmButton: true,

confirmButtonText: "Cerrar"

}).then(function(result){

if (result.value){

window.location = "adcatPr";

}

})

</script>';

}

}

}

Método ElminarCP



Img 4.10 método EliminarCP

static public function EliminarCP(){

if(isset($\_POST['txt\_id'])){

if(preg\_match('/^[1-9][0-9]\*$/',$\_POST['txt\_id'])){

$tabla = "categoria";

$data = array("Id" => $\_POST["txt\_id"]);

$res = ModeloGuardarP::mdlEliminarCP($tabla,$data);

if($res = "active"){

echo'<script>

Swal.fire({

icon: "success",

title: "Se ha eliminado el registro exitosamente!",

showConfirmButton: true,

}).then(function(result){

if (result.value){

window.location = "adcatP"

}

})

</script>';

}else{

echo'<script>

Swal.fire({

icon: "error",

title: "Error al aliminar el registro!",

showConfirmButton: true,

})

</script>';

}

}else{

echo'<script>

swal.fire({

icon: "error",

title: "Solo se permite números o hay datos vacíos!",

showConfirmButton: true,

confirmButtonText: "Cerrar"

}).then(function(result){

if (result.value){

window.location = "elcatP"

}

})

</script>';

}

}

**Método ElminarPr**

****

**Img 4.11 método EliminarPr**

**static public function EliminarPr(){**

**if(isset($\_POST['txt\_id'])){**

**if(preg\_match('/^[1-9][0-9]\*$/',$\_POST['txt\_id'])){**

**$tabla = "proveedor";**

**$data = array("Id" => $\_POST["txt\_id"]);**

**$res = ModeloGuardarP::mdlEliminarPr($tabla,$data);**

**if($res = "active"){**

**echo'<script>**

**Swal.fire({**

**icon: "success",**

**title: "Se ha eliminado el registro exitosamente!",**

**showConfirmButton: true,**

**}).then(function(result){**

**if (result.value){**

**window.location = "elcatPr"**

**}**

**})**

**</script>';**

**}else{**

**echo'<script>**

**Swal.fire({**

**icon: "error",**

**title: "Error al aliminar el registro!",**

**showConfirmButton: true,**

**})**

**</script>';**

**}**

**}else{**

**echo'<script>**

**swal.fire({**

**icon: "error",**

**title: "Solo se permite números o hay datos vacíos!",**

**showConfirmButton: true,**

**confirmButtonText: "Cerrar"**

**}).then(function(result){**

**if (result.value){**

**window.location = "elcatPr"**

**}**

**})**

**</script>';**

**}**

**}**

**}**

**}**