



Diplomado virtual en
PROGRAMACIÓN EN PHP
Guía didáctica 1: Conceptos básicos



Formación Virtual

.....educación sin límites



Competencia específica

Se espera que, con los temas abordados en la guía didáctica del módulo 1: Conceptos básicos, el estudiante logre la siguiente competencia específica:

- Conocer los conceptos básicos y aplicaciones del lenguaje de PHP para la programación en cuanto a variables, tipos de datos y operadores.



Contenidos temáticos

Los contenidos temáticos, para desarrollar en la guía didáctica del módulo 1: Conceptos básicos, son:

- 1 Conceptos básicos
- 2 Instalación de PHP - XAMMP
- 3 Sublime Text (editor de texto)
- 4 Variables
- 5 Tipos de datos

Ilustración 1: caracterización de la guía didáctica¹.

Fuente: autor.

¹ Algunas ilustraciones de esta guía no llevarán nombre, solo la fuente, ya que la mayoría de ellas corresponden a ejemplos hechos por el mismo docente del diplomado y otras buscan ser referencias visuales e ilustrativas, pero en todas se reconocerá la fuente.

Tema 1: Conceptos Básicos

Para entrar en materia de conceptos tan abstractos, y difíciles de definir en algunas ocasiones, es importante partir desde las bases características del área. Hay conceptos o términos que en un principio sonarán extraños y difíciles de comprender, pero poco a poco, a medida que se desarrollen los contenidos temáticos del diplomado, serán mejor comprendidos. ¡Así que ánimo!

Desde lo más básico: ¿qué es la programación?

El término de programación se reconoce como el proceso por el cual se diseña, codifica, limpia y protege el código fuente de programas informáticos. De acuerdo con ello el código se escribe, se prueba y se perfecciona para llegar a un desarrollo a la medida.

La programación se guía por una serie de reglas y un conjunto pequeño de órdenes, instrucciones y expresiones que tienden a parecerse a una lengua natural, pero en forma acotada. Los lenguajes de programación son todas aquellas reglas o normas, símbolos y palabras particulares empleadas para la creación de un programa que busca dar una solución a un problema determinado.



```
if ($(window).scrollTop() > header1_initialDistance) {  
  if (parseInt(header1.css('padding-top'), 10) == header1_initialPadding) {  
    header1.css('padding-top', '' + $(window).scrollTop() - header1_initialDistance + header1_initialPadding + 'px');  
  }  
} else {  
  header1.css('padding-top', '' + header1_initialPadding + 'px');  
}  
  
if ($(window).scrollTop() > header2_initialDistance) {  
  if (parseInt(header2.css('padding-top'), 10) == header2_initialPadding) {  
    header2.css('padding-top', '' + $(window).scrollTop() - header2_initialDistance + header2_initialPadding + 'px');  
  }  
} else {  
  header2.css('padding-top', '' + header2_initialPadding + 'px');  
}
```

Ilustración 2.
Fuente: Pixabay.

Ahora, entrelazando la definición de programación, es necesario responder al siguiente interrogante:

¿Qué es un lenguaje de programación?

Un lenguaje de programación es un lenguaje formal que especifica una serie de instrucciones (comando – códigos) para que una computadora produzca y procese diversas clases de datos. Los lenguajes de programación pueden usarse para crear programas que lleven a la práctica algoritmos específicos, los cuales controlan el comportamiento físico y lógico de una computadora. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación. («Lenguaje de programación», 2019)

```
<!DOCTYPE html>
<html>
<head>
  <title>PHP Test</title>
</head>
<body>
  <?php echo '<p>Hello Word</p>'; ?>
</body>
</html>
```

Ilustración 3.
Fuente: autor.

Ahora sí, es hora de entrar en materia.

¿Qué es PHP?

La respectiva documentación de PHP es muy directa al definirse: «PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML» (The PHP Group, s.f.).

PHP es el lenguaje de programación más extendido en la *web* y creado inicialmente en 1994. Se trata de un lenguaje de creación relativamente reciente, aunque debido a la rapidez con la que evoluciona internet parezca que ha existido toda la vida.

Es un lenguaje que ha tenido una gran aceptación en la comunidad de desarrolladores debido a la potencia y simplicidad que lo caracteriza, así como al soporte generalizado en la mayoría de los servidores de *hosting*, hasta los más simples y económicos.

La facilidad de PHP se basa en que permite implementar pequeños fragmentos de código dentro de lo que sería una página común creada con HTML. Con esos *scripts* PHP permite realizar determinadas acciones de una forma fácil y eficaz, pudiendo realizar todo tipo de tareas, desde las más simples hasta las más complejas.

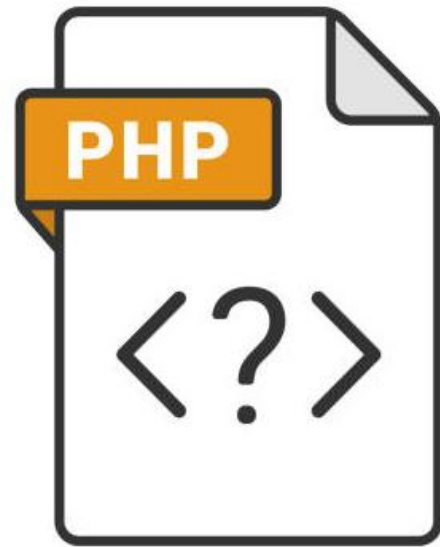


Ilustración 4.
Fuente: iStockphoto.

En resumen, con PHP se escriben *scripts* dentro del código HTML. Como ya se está familiarizado con HTML, empezar a desarrollar con PHP es prácticamente inmediato. Por otra parte, y es aquí donde reside su mayor interés, PHP ofrece un sinnúmero de funciones para la explotación de todo tipo de recursos, entre los que destacan las bases de datos, a las que se podrá acceder de una manera llana y sin complicaciones.

PHP cuenta con unas características muy bien impartidas desde sus fundamentos, las cuales le permiten mantener el estándar de ser uno de los lenguajes de programación más usados en la actualidad, entre las principales características se encuentran las siguientes:

✓Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.

✓Se considera lenguaje de programación fácil de aprender.

✓El código fuente escrito en PHP es invisible al navegador web.

✓Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad y entre ellos se destaca su conectividad con MySQL.

✓Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).

✓Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

✓Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

Permite aplicar técnicas de programación orientada a objetos.
Debido a su flexibilidad ha tenido una gran acogida como lenguaje base para las aplicaciones web de manejo de contenido, y es su uso principal.

Ilustración 5: características de PHP.

Fuente: «PHP» (2022).

¿Qué puede hacer PHP?

- ✓ Generar contenido dinámico de páginas web.
- ✓ Crear, abrir, leer, escribir, eliminar y cerrar archivos en el servidor.
- ✓ Recopilar datos de formularios.
- ✓ Agregar, eliminar, actualizar y leer datos de una base de datos, valga la redundancia.



¿Te interesa conocer un poco más sobre PHP? te invito a leer el historial de lanzamientos de sus respectivas versiones, disponible en: https://es.wikipedia.org/wiki/PHP#Historial_de_lanzamiento

¿Qué es HTML?

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de internet. Se trata de las siglas que corresponden a *hypertext markup language*, es decir, lenguaje de marcas de hipertexto.



Ilustración 6.
Fuente: iStockphoto.

Tema 2: Instalación de PHP - XAMPP

Ahora que conoces un poco el significado de PHP y algunos conceptos que engloban lo que representa, se puede proceder a la descarga e instalación de las herramientas necesarias para el correcto desarrollo del diplomado, principalmente **XAMPP**.

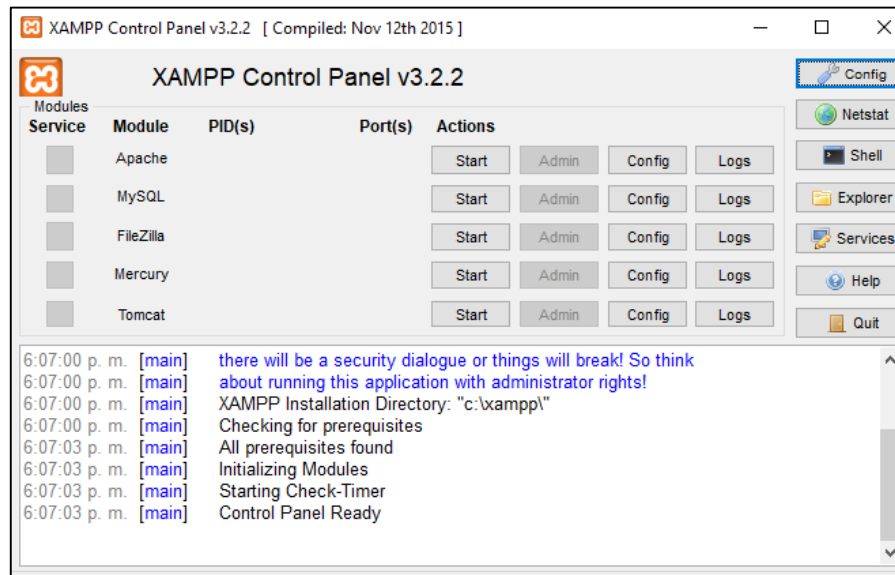


Ilustración 7.
Fuente: autor.

¿Qué es XAMPP?

XAMPP es un paquete de *software* libre que hará el papel de servidor para hacer el respectivo uso de PHP, consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes PHP (Apache Friends, s.f.).

Dentro de todas las funciones y características que se destacan en XAMPP hay unas que se deben tener en cuenta y son importantes conocer:

- ✓ Para Windows existen dos versiones, una con instalador y otra portable (comprimida) para descomprimir y ejecutar.
- ✓ Otra característica, no menos importante, es que la licencia de esta aplicación es GNU (*general public license*), está orientada principalmente a

proteger la libre distribución, modificación y uso de *software*. Su propósito es declarar que el *software* cubierto por esta licencia es *software* libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

- ✓ XAMPP solamente requiere descargar y ejecutar un archivo —.zip, .tar, o .exe— con unas pequeñas configuraciones en alguno de sus componentes que el servidor web necesitará.
- ✓ Una de las características sobresalientes de este sistema es que es multiplataforma, es decir, existen versiones para diferentes sistemas operativos, tales como: Microsoft Windows, GNU/Linux, Solaris, y MacOS X. Existen versiones para Linux (testado para SuSE, RedHat, Mandrake y Debian), Windows (Windows 98, NT, 2000, XP y Vista), MacOS X y Solaris (desarrollada y probada con Solaris 8, probada con Solaris 9).
- ✓ XAMPP se actualiza regularmente para incorporar las últimas versiones de Apache/MySQL/PHP y Perl. También incluye otros módulos como OpenSSL y phpMyAdmin.



XAMPP es una herramienta increíble y fácil de utilizar en el desarrollo con PHP; será la herramienta madre que utilizaremos en el diplomado para desarrollo en PHP. Te invito a que conozcas un poco más de ella y, más importante aún, cómo instalarla.

A continuación, comparto un video que explica el proceso de descarga e instalación:

<https://www.youtube.com/watch?v=BdqsA0DvIZI>.

O directamente en el enlace: <https://www.apachefriends.org/es/index.html>, siguiendo el respectivo paso a paso intuitivo que presenta para su instalación («siguiente», «siguiente», «aceptar», etc.), pero ten en cuenta el sistema operativo con el que cuenta el PC, ya sea Windows, MacOS o Linux:

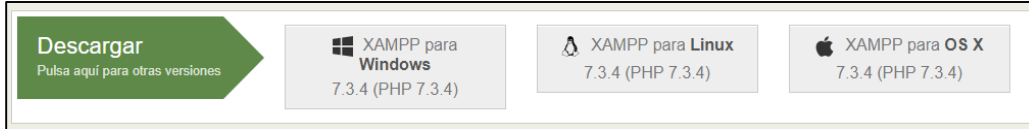


Ilustración 8.

Fuente: autor.

¿Cómo reconocer que XAMPP está correctamente instalado?

Haciendo hincapié en la facilidad de descarga e instalación, se reconoce el correcto funcionamiento de XAMPP cuando dos de las características fundamentales (MySQL y Apache) de la herramienta se ejecutan sin ningún problema dando clic en el botón de «**Start**».

Las demás herramientas y opciones son nativas y funcionales de XAMPP, para efectos del diplomado solo será foco de estudio MySQL (motor de bases de datos) y Apache (servidor de ejecución de PHP).

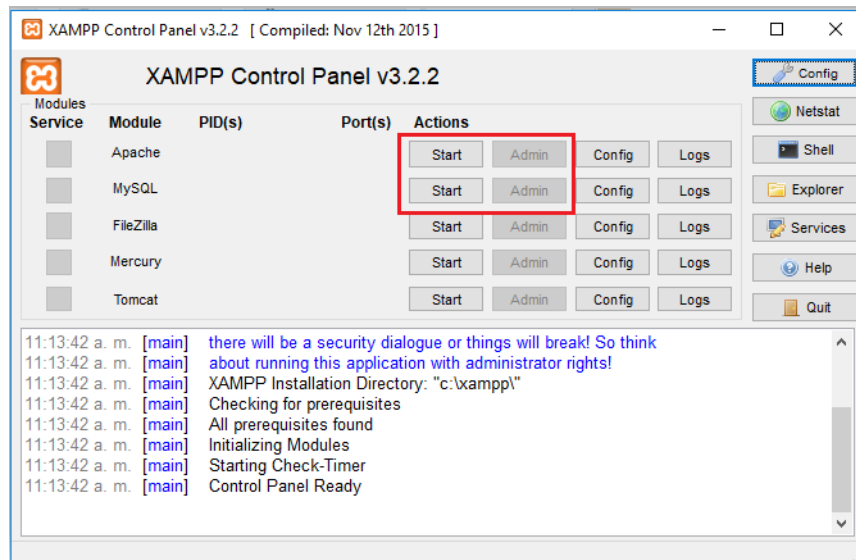


Ilustración 9.

Fuente: autor.

El resultado de la acción anteriormente mencionada es la correcta ejecución del panel y de los procesos seleccionados (MySQL y el servidor de MySQL, Apache).

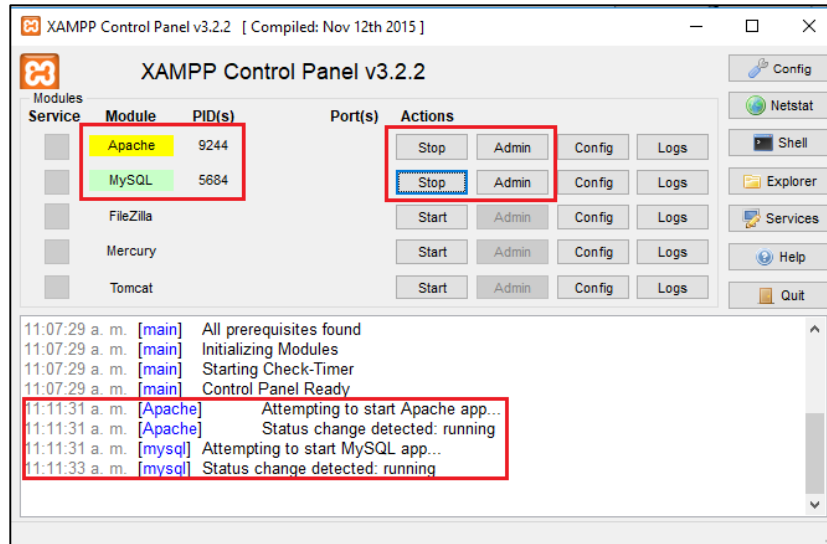


Ilustración 10.

Fuente: autor.

XAMPP presenta algunos problemas con alguno de los puertos que utiliza para funcionar dado que determinados programas utilizan los mismos puertos, por ejemplo Skype, el más común de todos; por ende, cuando suceda este problema es recomendable cerrar Skype o la herramienta o programa que esté presentado problemas y reiniciar XAMMP.

Además, recuerda que cualquier duda, inquietud o dificultad será atendida a través del correo: diegovalencia@politecnicodecolombia.edu.co

Tema 3: Sublime Text (Editor de Texto)

Ya se cuenta con la herramienta de ejecución de PHP, ahora hace falta un editor de texto para plasmar las líneas de codificación del lenguaje; existen diversas herramientas para codificar, entre ellas Sublime Text, NotePad++, Atom, NetBeans, Eclipse, Dreamweaver e, incluso, el bloc de notas sirve para codificar. Para el desarrollo del curso haremos uso de Sublime Text, pero no quiere decir que no se pueda usar otra herramienta, el estudiante es libre de elegirla, siempre y cuando tenga conocimiento en cuanto al manejo de esta.

¿Qué es Sublime Text?

Sublime Text es un editor de texto y editor de código fuente multiplataforma. Es una herramienta concebida para programar sin distracciones. Su interfaz de color oscuro y la riqueza de coloreado de la sintaxis centra nuestra atención completamente.



Ilustración 11.
Fuente: Pixabay.

Entre las principales características de Sublime Text, resaltan en gran medida las siguientes:

- ✓ **Minimapa:** consiste en una previsualización de la estructura del código, es muy útil para desplazarse por el archivo cuando este es muy extenso.
- ✓ **Multi selección:** hace una selección múltiple de un término por diferentes partes del archivo.

- ✓ **Multi cursor:** crea cursores con los que se puede escribir texto de forma arbitraria en diferentes posiciones del archivo.
- ✓ **Multi layout:** trae siete configuraciones de plantilla, de las cuales se puede elegir editar en una sola ventana o hacer una división de hasta cuatro ventanas verticales o cuatro ventanas en cuadrícula.
- ✓ **Soporte nativo para infinidad de lenguajes:** soporta de forma nativa 43 lenguajes de programación y texto plano.
- ✓ **Búsqueda dinámica:** se puede hacer búsqueda de expresiones regulares o por archivos, proyectos, directorios, una conjunción de ellos o todo a la vez.
- ✓ **Auto completado y marcado de llaves:** se puede ir a la llave que cierra o abre un bloque de una forma sencilla.
- ✓ **Configuración total de *keybindings*:** todas las teclas pueden ser sobrescritas al gusto.
- ✓ **Coloreado y envoltura de sintaxis:** si se escribe en un lenguaje de programación o marcado, resalta las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.
- ✓ **Pestañas:** se pueden abrir varios documentos y organizarlos en pestañas.
- ✓ **Resaltado de paréntesis e indentación:** cuando el usuario coloca el cursor en un paréntesis, corchete o llave resalta esta y el paréntesis, corchete o llave de cierre o apertura correspondiente (Sublime Text, 2019).

Lenguajes soportados por Sublime Text					
ActionScript	Clojure	Graphviz (DOT)	Lua	Perl	Scala
Apple Script	CSS	Groovy	Lisp	PHP	Shell Script (Bash)
ASP	D	Haskell	Makefile	Python	SQL
Batch File	Diff	HTML	Markdown	R	Td
C	Erlang	Java	MATLAB	Rails	Texto plano
C#	Expresión regular	JavaScript	Objective-C	ReStructuredText	Textile
C++	Go	LaTeX	Ocaml	Ruby	XML
XSL	YAML				

Tabla 1: lenguajes soportados por Sublime Text.

Fuente: autor.

¿Cómo descargar Sublime Text?

Sublime Text presenta para Windows dos tipos de versiones, la versión completa y la versión portable, la recomendación es hacer uso de la versión completa; ten muy en cuenta el sistema operativo al igual que con XAMPP, y más aún en la arquitectura, sí será de x32 o x64 bits (para el caso de Windows en especial).



Para hacer efectiva la descarga de Sublime Text correctamente, debe ingresar al portal de la plataforma. disponible en <https://www.sublimetext.com/3>

Version: Build 3207

- [OS X](#) (10.7 or later is required)
- [Windows](#) - also available as a [portable version](#)
- [Windows 64 bit](#) - also available as a [portable version](#)
- [Linux repos](#) - also available as a [64 bit](#) or [32 bit tarball](#)

Ilustración 12.

Fuente: autor.

Luego de la descarga el proceso de instalación es muy sencillo, consiste en un paso a paso de «siguiente» y «aceptar». Con lo anterior se tendrá ya descargada e instalada la herramienta de codificación para el desarrollo del diplomado, recuerda que cualquier otra herramienta es perfectamente válida, siempre y cuando se conozca la forma de usarse.

Hola Mundo



Ya que conoces un poco lo qué es PHP, tienes instaladas las herramientas necesarias; se procede a realizar un primer programa en el lenguaje de programación PHP. En este caso será el famoso «Hola Mundo», conocido por ser las primeras líneas de código de un desarrollador de software en cualquier lenguaje. Donde adicionalmente se identificarán las primeras características que brinda el lenguaje de programación a la hora de ser empleado y la configuración inicial que se ocupa para el desarrollo.

Para iniciar a codificar en PHP habrá que hacer uso de las dos herramientas mencionadas en los temas anteriores: XAMPP y Sublime Text, nuestro servidor para ejecutar PHP y el editor de texto, respectivamente. Al ser PHP un lenguaje de programación orientado a la web, nuestra consola o lugar de ejecución o visualización de resultados será el navegador, sea Chrome, Opera, Microsoft Edge u otro, el código será ejecutado y visualizado en estos.

El primer paso es identificar dónde quedó instalado XAMPP, la ruta de instalación se puede observar al momento de instalarse la herramienta o en la mayoría de los casos se encuentra en el disco local (C), exactamente en la siguiente ruta: **C:\xampp**

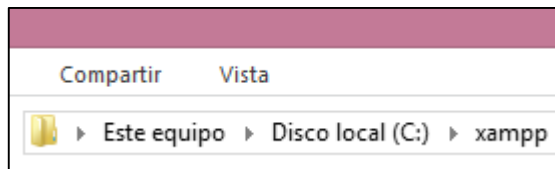


Ilustración 13.

Fuente: autor.

Ya teniendo ubicada la ruta de instalación, dentro del directorio de carpetas hay una en especial que será la usada a lo largo del diplomado: **htdocs**.

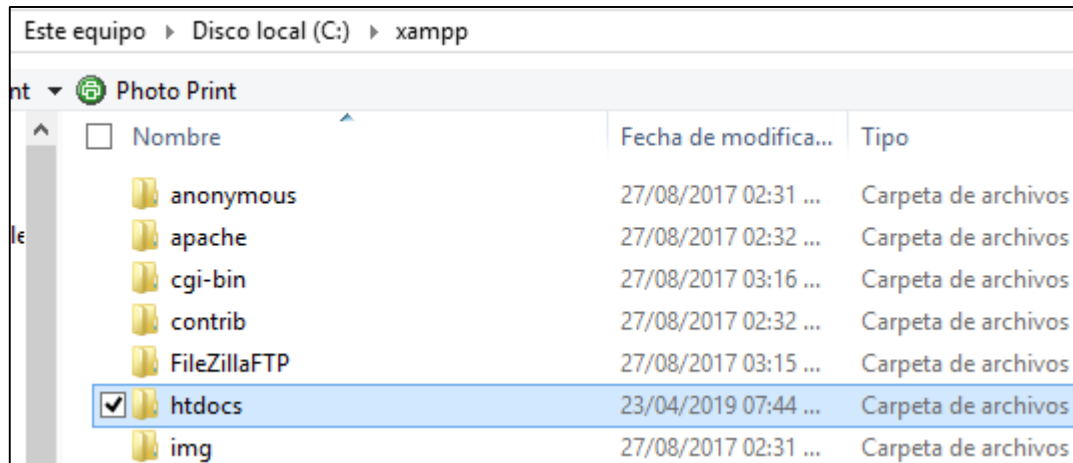
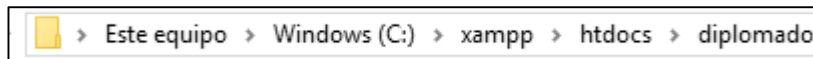


Ilustración 14.
Fuente: autor.

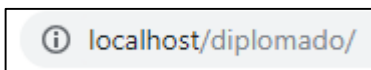
Cuando se realiza una petición al servidor este comienza a buscar el *path* que se le indique desde la raíz del sitio; para el caso de XAMPP es la carpeta *htdocs*, que está localizada (de forma predeterminada) dentro de la carpeta de instalación de XAMPP (por ejemplo: *c:\xampp*).

Según lo anterior, por ejemplo, en caso de contar con un proyecto de nombre «diplomado», las rutas de ubicación y de petición serían:

- ✓ Ubicación: **C:\xampp\htdocs\diplomado**



- ✓ Petición: **http://localhost/diplomado/**



Ilustraciones 15 y 16.
Fuente: autor.

Siendo la ubicación, en ese sentido, el lugar donde se encuentra almacenado el proyecto dentro del servidor de XAMPP y la ruta de petición, la indicada para abrir el proyecto en el navegador por medio de la URL.

Nota: para abrir un proyecto a través de la ruta de petición del navegador (<http://localhost/NOMBRE DEL PROYECTO>) **siempre** se debe realizar cuando los servicios de XAMPP estén activos.

En resumen, htdocs será el directorio encargado de contener todos los proyectos y el lugar que el servidor buscará cuando se realice peticiones al mismo, además, es importante resaltar que no solo se puede realizar un proyecto a la vez, por el contrario, se puede contener los proyectos que se deseen, como en este caso:

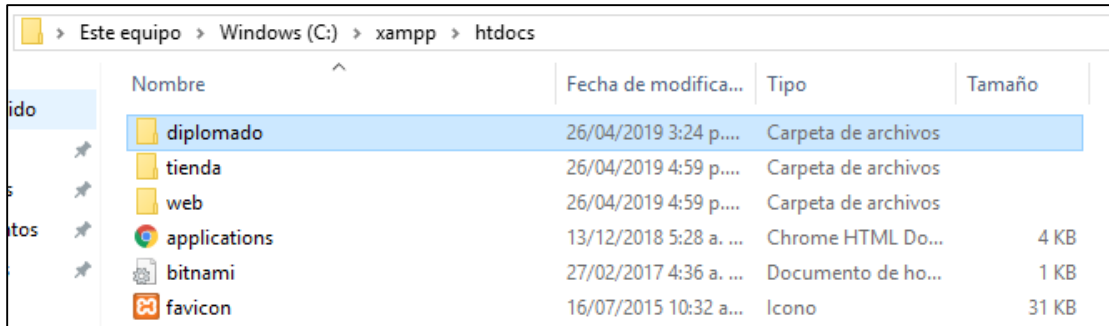


Ilustración 17.
Fuente: autor.

En la carpeta de htdocs es muy constante encontrar otros archivos, como en se muestra en la imagen anterior (*applications, bitnami, favicon, etc.*); son archivos que por defecto XAMPP almacena en htdocs, así que se puede hacer caso omiso de estos.

La estructura que se manejará a lo largo del diplomado estará compuesta por un proyecto, el cual contendrá un «subproyecto» con el contenido de cada tema que se vaya desarrollando dentro del diplomado, siendo el primero de los temas el «Hola mundo», de la siguiente forma:

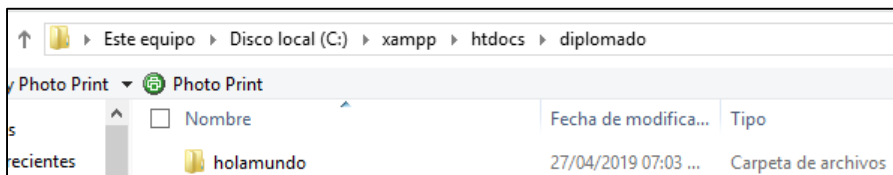


Ilustración 18.
Fuente: autor.

Teniendo un directorio, carpeta principal o nombre del proyecto: «diplomado» (mucho cuidado con las mayúsculas y minúsculas, en la medida de lo posible la notación de todo será en minúsculas), y dentro de esta cada uno de los temas, logrando así la siguiente ruta de ubicación: C:\xampp\htdocs\diplomado\holamundo

y la ruta de petición: <http://localhost/diplomado/holamundo/> (recordar siempre activar los servicios de XAMPP y que estos estén funcionando bien).

Hasta este punto se encuentra lista toda la estructura del proyecto, hace falta un último paso para iniciar la codificación de lleno. Cuando el servidor de XAMPP recibe una petición, aparte de buscar el directorio o proyecto solicitado, este buscará un archivo «index», los cuales son archivos que PHP busca y ejecuta automáticamente sin necesidad de la escritura de su nombre dentro de la petición. Cuando se realizan peticiones, como por ejemplo <http://localhost/diplomado/holamundo/>, el resultado en el navegador será el siguiente:

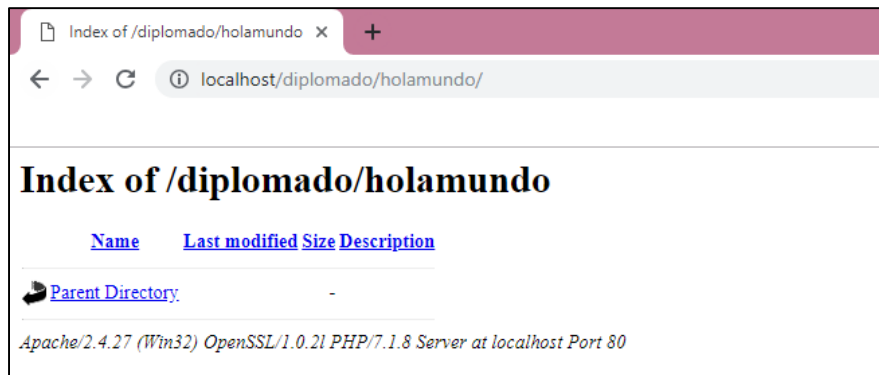


Ilustración 19.
Fuente: autor.

En este caso se encuentra vacío y PHP mostrará el contenido del directorio, en caso de encontrar carpetas o archivos estos serán listados así:

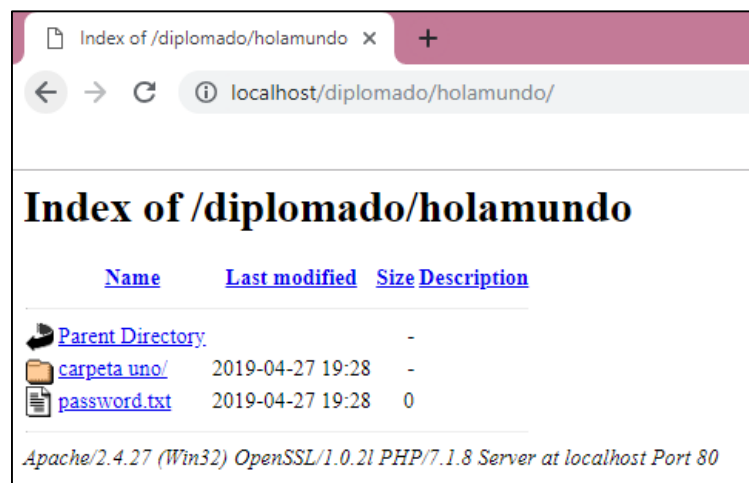


Ilustración 20.

Fuente: autor.

Con la misma petición al mismo directorio, PHP lista en este caso la información que se encuentra contenida en este; otro ejemplo sería una petición a directorio de «diplomado», en este orden de ideas, deberá listar dentro de su directorio a «holamundo», véase:

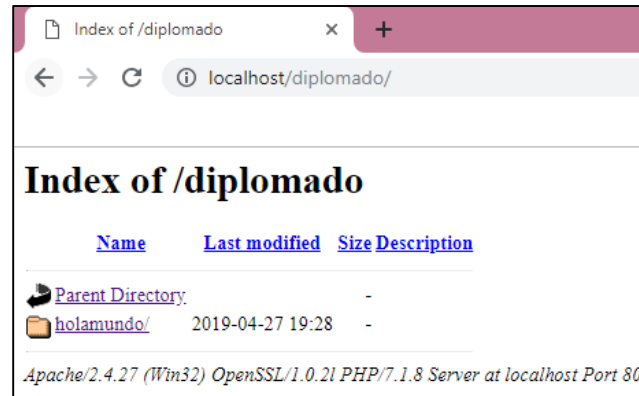


Ilustración 21.

Fuente: autor.

Todo esto sucede ya que el servidor y la petición no encuentran un archivo de ejecución y simplemente visualizan el contenido de los directorios, ya en este caso entra en juego los archivos con extensión .PHP, que serán los archivos en los que se ejecuten automáticamente las peticiones y más específicamente PHP, en primera parte los index.php.

Entendido todo lo anterior, ahora se hará uso del editor de texto seleccionado, al abrirlo la interfaz que se observará será la siguiente:

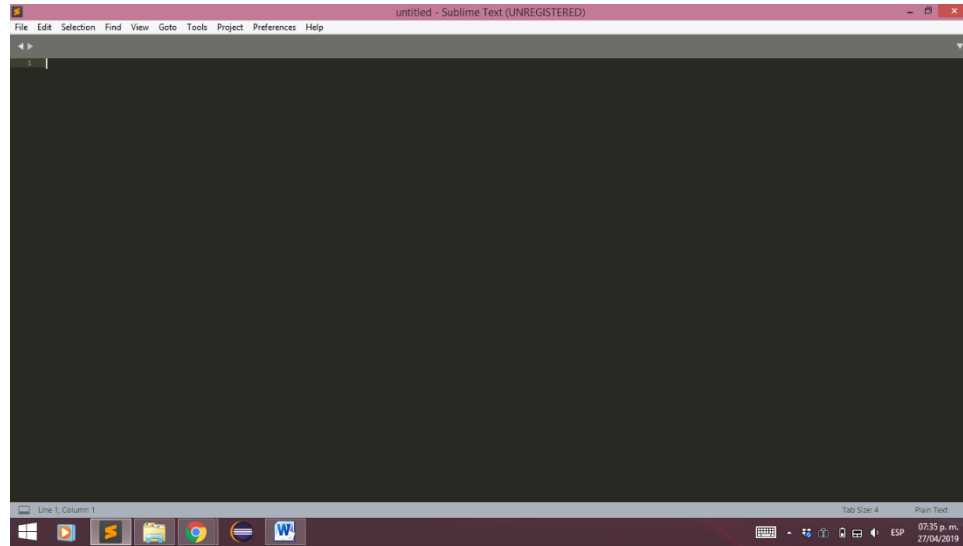


Ilustración 22.

Fuente: autor.

El siguiente paso es abrir o llevar el proyecto «diplomado» a Sublime Text, hay dos formas de hacerlo:

- ✓ Arrastrar la carpeta/proyecto/directorio directamente a Sublime Text.
- ✓ Ir a «file/archivo» dentro del menú y seleccionar «open folder/abrir carpeta», luego buscar respectivamente la ubicación de esta en: **C:\xampp\htdocs**.
- ✓ El resultado debe ser similar al siguiente luego de abrir respectivamente el proyecto:

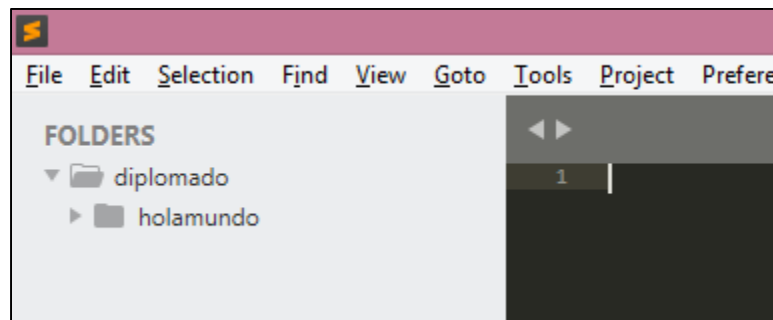


Ilustración 23.

Fuente: autor.

A lo largo del diplomado y a medida que se vayan desarrollando los conceptos se irá haciendo uso de las funciones principales que presta Sublime Text como editor de texto.

El paso para crear el archivo index.php, o cualquier archivo en su defecto, es el siguiente:

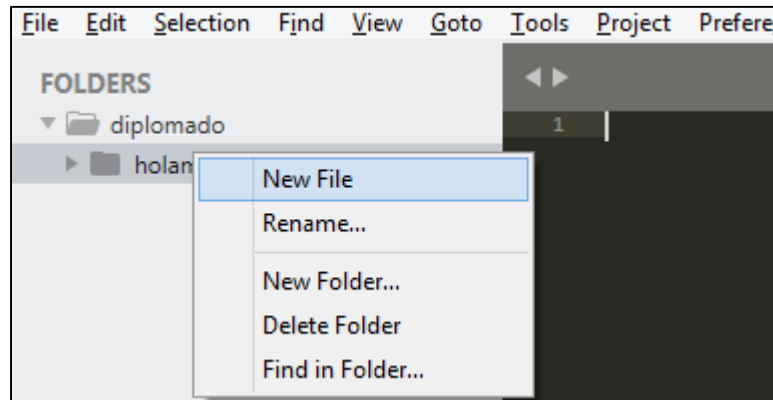


Ilustración 24.

Fuente: autor.

Lo primero es identificar la carpeta/directorio donde se desea crear el archivo, dado que de esta forma quedará contenido en él. El proceso se realiza dando clic derecho y seleccionando la opción de «New File», esta opción invocará una pestaña «sin título».

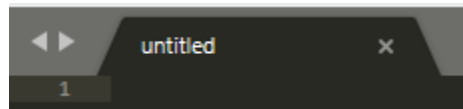


Ilustración 25.

Fuente: autor.

Para nombrar esta se utilizará un comando que será muy usado dentro de Sublime Text.

CONTROL [CTRL] + S

El comando **control + S** indica a Sublime Text que se desea guardar algo, ya sea un archivo recién creado o un cambio hecho dentro de un archivo. En el presente caso será para nombrar y guardar el archivo, que tendrá el nombre de index.php.

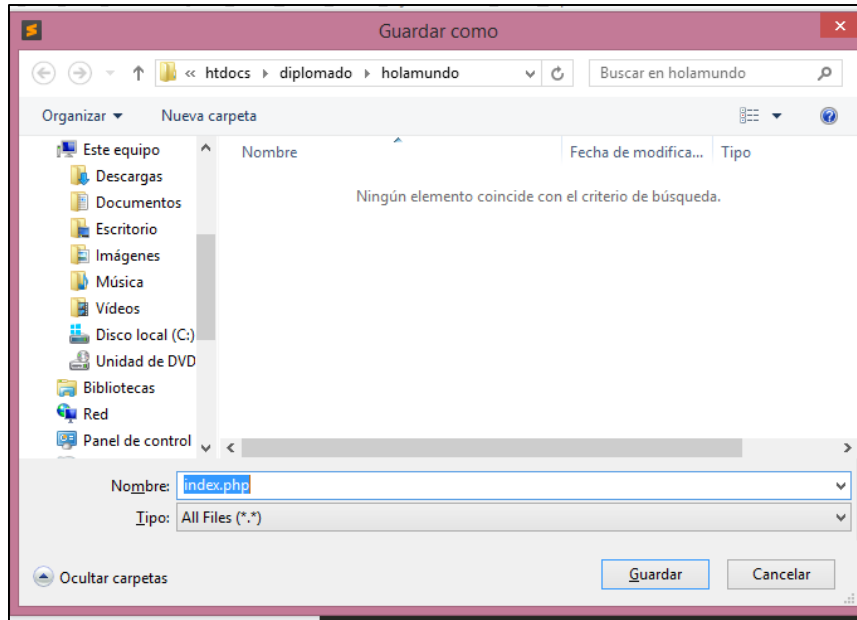


Ilustración 26.
Fuente: autor.

El resultado de la creación del archivo index.php se puede observar en dos partes, la pestaña que anteriormente estaba sin título y en el directorio/carpeta del proyecto donde se decidió crear:

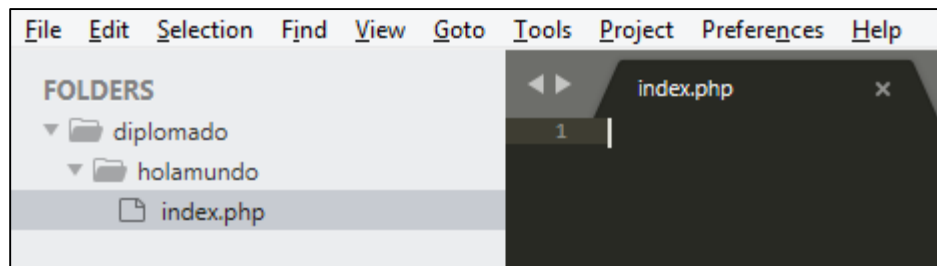


Ilustración 27.
Fuente: autor.

Además, una prueba que se puede realizar es ir al navegador y realizar una petición a <http://localhost/diplomado/holamundo/>, donde no se obtendrá el resultado de la siguiente imagen sino que será el contenido que presente el archivo index.php, que es el que se ejecuta por defecto dentro de «holamundo». Es importante recordar que estos directorios o proyectos en PHP siempre buscan un

archivo index (sea PHP o HTML en algunos casos) para ser ejecutado, en caso contrario ejecuta el contenido de la siguiente imagen:

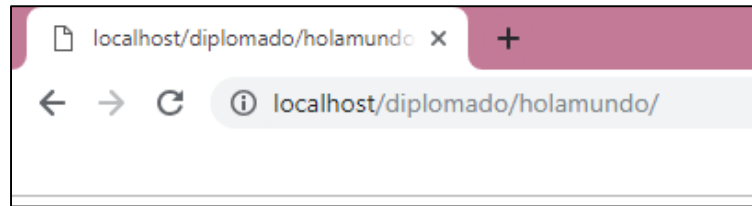


Ilustración 28.

Fuente: autor.

Como se mencionaba anteriormente, PHP busca automáticamente dentro de las peticiones a directorios el archivo index.php y lo ejecuta sin indicar «index.php» en la URL del navegador. Pero ¿qué sucede en caso de que se realice la petición indicando directamente en la URL «index.php»?

<http://localhost/diplomado/holamundo/index.php>:

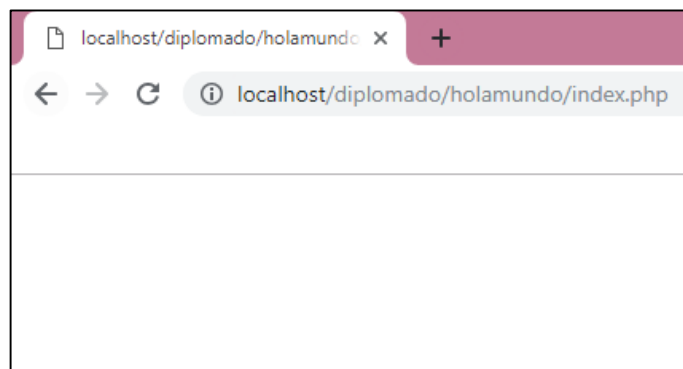


Ilustración 29.

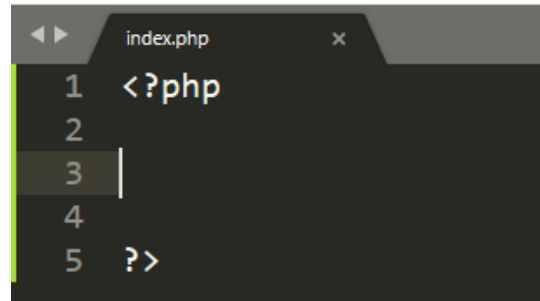
Fuente: autor.

Nota: el mismo resultado se puede obtener al incluir un archivo index.html dentro de la petición o, en caso de no existir un archivo index.php pero sí un index.html, y además en la petición no se incluye, PHP automáticamente ejecutará el index.html.

El resultado es exactamente el mismo, una página completamente en blanco sin señal de errores. Ahora sí, en materia de codificación se puede empezar.

Para la escritura bajo el lenguaje de PHP él analiza un fichero/archivo, en este caso index.php, busca las etiquetas de apertura y cierre, que son `<?php` y `?>` (respectivamente) y que indican a PHP dónde empezar y finalizar la interpretación

o lectura del código. Este mecanismo permite embeber a PHP en todo tipo de documentos, ya que todo lo que esté fuera de las etiquetas de apertura y cierre de PHP será ignorado por el analizador.



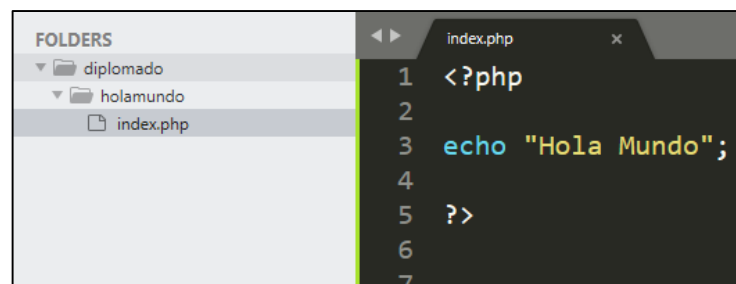
```
index.php x
1 <?php
2
3
4
5 ?>
```

Ilustración 30.
Fuente: autor.

Nota: recuerda que después de realizar cada cambio se debe realizar el guardado con el comando CONTROL + S para que todos estos sean guardados con éxito.

Esa es la sintaxis básica que siempre se debe usar al momento de trabajar con PHP. Con todo preparado falta escribir el mensaje que incumbe en este momento: «Hola mundo». Para realizar esto, hay que utilizar el primer comando/función de PHP que servirá para imprimir mensajes, y este es **echo**.

echo aunque directamente no es una función, sino una construcción del lenguaje, permite básicamente imprimir información en el navegador. Existe otra construcción del lenguaje de igual aplicación que **echo**, la cual es **print**, pero para este caso solo se hará uso de **echo**. Además de hacer uso de este, debe ir acompañado del mensaje que se desea imprimir y que debe ir en comillas inglesas dobles (" ").



```
FOLDERS
  diplomado
  holamundo
  index.php

index.php x
1 <?php
2
3 echo "Hola Mundo";
4
5 ?>
6
7
```

Ilustración 31.
Fuente: autor.

Hay varias características a tener en cuenta con la escritura del «hola mundo» y son las siguientes: las construcciones del lenguaje, funciones o demás agregaciones, deben ir separadas por una espacio de la próxima acción/instrucción que se desea realizar, por el momento las comillas dobles serán las que se usarán a la hora de imprimir mensajes y por último, y más importante, toda instrucción/renglón de PHP debe terminar en punto y coma «;» (hay excepciones como la declaración de funciones, condicionales, ciclos y demás, más adelante en el diplomado se irán conociendo).

Para ejecutar y visualizar el resultado de «Hola Mundo» dentro del navegador, lo primero es guardar los cambios, recordar siempre el uso de control + S e ir al navegador y volver a realizar la petición, o simplemente presionar F5 para obtener los resultados dentro del navegador.

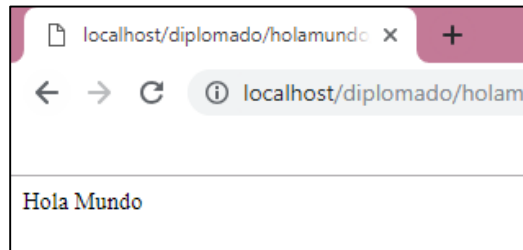


Ilustración 32.

Fuente: autor.

En resumen, «Hola Mundo» ha sido el primer proyecto o las primeras líneas dentro de PHP, con este se han configurado todas las herramientas necesarias para el adecuado ambiente de desarrollo (XAMPP y Sublime Text), además de haber tenido en cuenta las características importantes en la creación de proyectos, directorios y archivos. Se conoció precisamente donde se deben almacenar estos y por cual medio se pueden ejecutar.

Para tener en cuenta:

- ✓ Identificar correctamente los servicios de XAMPP y cuando estos están activos o no.
- ✓ Localizar la carpeta de instalación de XAMPP y htdocs.
- ✓ Ser cuidadoso en el nombre de los archivos y carpetas.

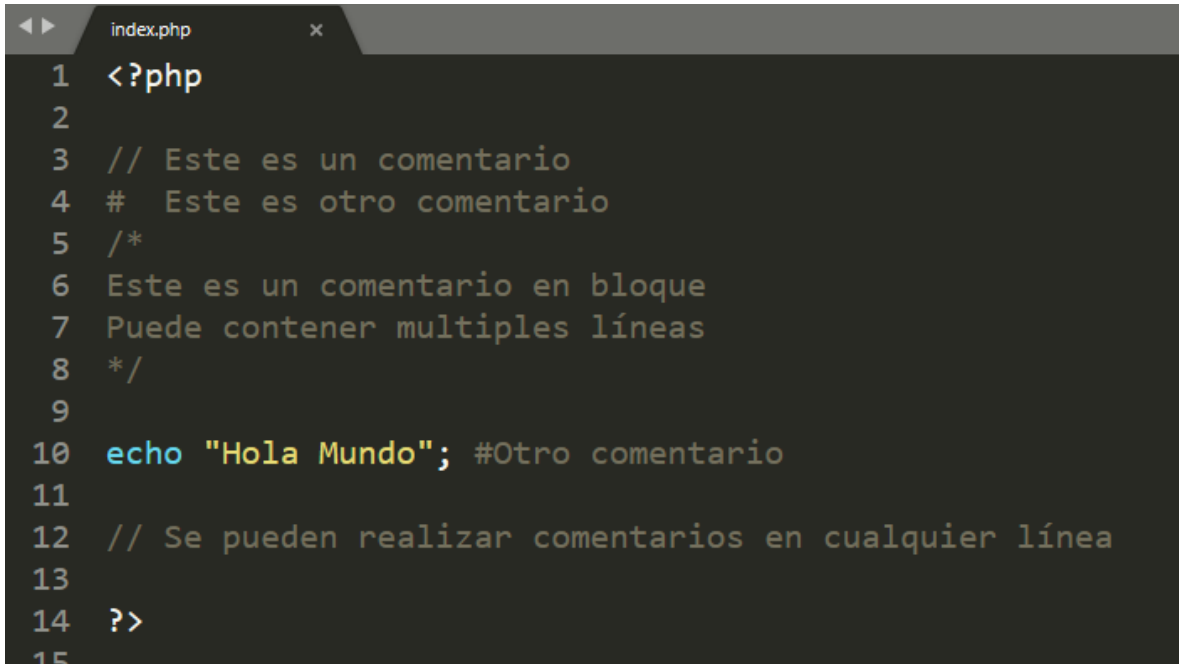
- ✓ PHP realiza una petición directa a los archivos index.php, pese a que en la petición no se encuentre este.
- ✓ echo y print cumplen la misma función. Para efectos de este diplomado solo se hará uso de echo.
- ✓ Las etiquetas de apertura y cierre de PHP son <?php y ¿>, respectivamente.
- ✓ Control + S permite guardar en Sublime Text.
- ✓ La dirección de petición es: <http://localhost/NOMBREDELPROYECTO>.
- ✓ Toda instrucción de PHP termina en punto y coma «;».

Comentarios

Un comentario en PHP es una línea que no es leída o ejecutada como parte del código/programa/proyecto. El propósito de esta línea es ser leída por otra persona que se encuentre realizando la lectura del código.

Estos comentarios son de gran importancia cuando se trabaja en un mismo código con X cantidad de personas, sirve principalmente para anotar o resaltar de forma directa fragmentos o información que puede ser relevante para otras personas o el mismo desarrollador, estos comentarios hacen las veces de bitácora dentro del código de los archivos .PHP.

Existencia varias formas de comentar, PHP soporta las siguientes:



```
1 <?php
2
3 // Este es un comentario
4 # Este es otro comentario
5 /*
6 Este es un comentario en bloque
7 Puede contener multiples líneas
8 */
9
10 echo "Hola Mundo"; #Otro comentario
11
12 // Se pueden realizar comentarios en cualquier línea
13
14 ?>
15
```

Ilustración 33: comentarios en PHP.

Fuente: autor.

En la ilustración anterior, los comentarios se encuentran plasmados sobre el mismo archivo `index.php` del proyecto «holamundo», para ver el resultado de este se debe recordar activar los servicios de XAMPP e ir al navegador a realizar la petición sobre el proyecto `holamundo/`, con o sin `index.php`.

Véase el resultado:

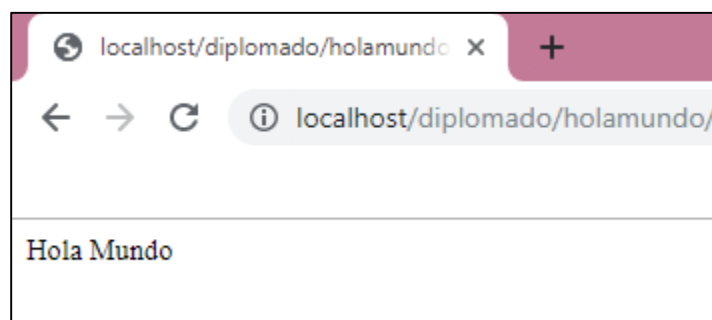


Ilustración 34.

Fuente: autor.

El resultado es el mismo a la primera ejecución del «Hola Mundo», sin comentarios; como se describe, los comentarios en todas sus formas de declaración no son tenidos en cuenta por el servidor, por lo que su único lugar de lectura se

encuentra dentro del código en los editores del texto y nunca serán tenidos en cuenta dentro del navegador.

Tema 4: Variables

Todo programa de ordenador o navegador persigue ofrecer una funcionalidad determinada para la que, por regla general, necesitará almacenar y manipular información. Dicha información son los datos sobre los que se opera y trabaja en un lenguaje de programación, deben almacenarse temporalmente en la memoria del ordenador/navegador. Para poder almacenar y recuperar fácilmente la información, los lenguajes de programación ofrecen el concepto de variables y PHP no es la excepción, que no son más que nombres que «apuntan» a una determinada parte de la memoria y que el lenguaje utiliza para escribir y leer datos de manera controlada.

El acceso a esta información depende del tipo de información, valga la redundancia, que se almacena. Por ejemplo, no es lo mismo tener la necesidad de manejar números, que letras; y dentro de estos no es igual tener que almacenar un número entero que uno decimal. Aunque al final todos son ceros y unos [0, 1], es la forma de interpretarlos lo que marca la diferencia, tanto al almacenarlos como al recuperarlos; PHP tiende a ser menos sensible en el manejo de los tipos de datos, para él «no existen» los tipos de datos, caso contrario de Java, por ejemplo, que tipifica muy bien los datos. En PHP una variable que almacena un número, en otra línea de código, puede pasar a ser una cadena de texto. Solo estas variables tendrán asignado un «tipo de dato» dependiendo del contexto, más adelante se explicará este concepto con mayor claridad.

En otras palabras, más cortas, las variables en PHP son contenedores para almacenar información indiferente de su tipo.

PHP es un lenguaje no tipado. Esto significa que las variables necesitan ser inicializadas y su tipo de dato no solo no precisa ser indicado, sino que este puede cambiar. PHP simplemente sabe el tipo de dato que se utiliza en cada momento dependiendo del contexto en que se utilice.

En PHP la forma de representar las variables es por medio del signo del dinero «\$», seguido del nombre de la variable.

Nota: al ser variables un nuevo tema, es ideal para el desarrollo del diplomado crear un nuevo proyecto o directorio dentro de «diplomado», igual al proceso que se realizó con «holamundo»; en este caso sería «variables» y de una vez su archivo index.php, así en Sublime Text:

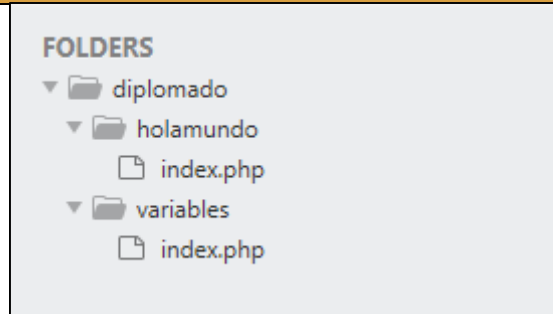


Ilustración 35.

Fuente: autor.

De forma tal que, en «diplomados», ahora existirán dos directorios/proyectos:

- ✓ **«holamundo»:** contiene las primeras líneas y proyecto de PHP.
 - Ubicación: **C:\xampp\htdocs\diplomado\holamundo**
 - Petición: <http://localhost/diplomado/holamundo/index.php> o <http://localhost/diplomado/holamundo/> (recordar que PHP puede ejecutar la petición sin index o con este).
- ✓ **«Variables»:** contendrá el código referente al uso, manejo y declaración de las variables en PHP.
 - Ubicación: **C:\xampp\htdocs\diplomado\variables.**
 - Petición: <http://localhost/diplomado/variables/index.php> o <http://localhost/diplomado/variables/> (recuerda que PHP puede ejecutar la petición sin index o con este).

De nuevo, y a forma de recordatorio, tener presente mantener activos los servicios de XAMPP.

Variables por valor

Ya dentro del index.php perteneciente a «variables» (cuidado al confundir los index.php).



Ilustración 36.

Fuente: autor.

En el anterior ejemplo se implementan tres variables:

- ✓ Número: contiene un número entero.
- ✓ Nombre: contiene una cadena de texto.
- ✓ Altura: contiene un número decimal.

De esta forma y a breves rasgos se identifican algunas cualidades de las variables, que son:

- ✓ Cuando se le asigna un valor de texto a una variable, esta debe ir acompañada de comillas alrededor del valor.

```
$nombre = "Diego";
```

Ilustración 37.

Fuente: autor.

- ✓ A las variables en PHP se les asignan valores, como en muchos otros lenguajes, con el símbolo de igual (=).
- ✓ A diferencia de otros lenguajes de programación, PHP no tiene comando para declarar una variable. Se crea en el momento en que primero se le asigna un valor. Por lo que no existirán declaraciones del siguiente tipo:

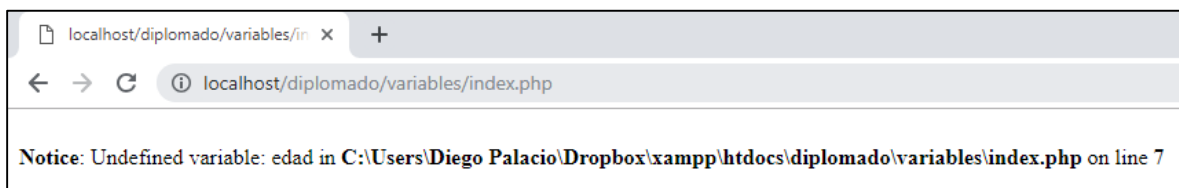
```
index.php
1  <?php
2
3  //Variables
4
5  $edad;
6
7  ?>
```

Ilustración 38.

Fuente: autor.

¿Qué quiere decir esto? Que la declaración de una variable solo se dará cuando se le asigne un valor; en el caso de la imagen anterior, efectivamente se declara una variable, pero al momento de la declaración no cuenta con ningún valor asignado o función, la forma de comprobar eso es realizando un «echo» a esa variable, de la siguiente forma:

```
1  <?php
2
3  //Variables
4
5  $edad;
6
7  echo $edad;
8
9  ?>
```



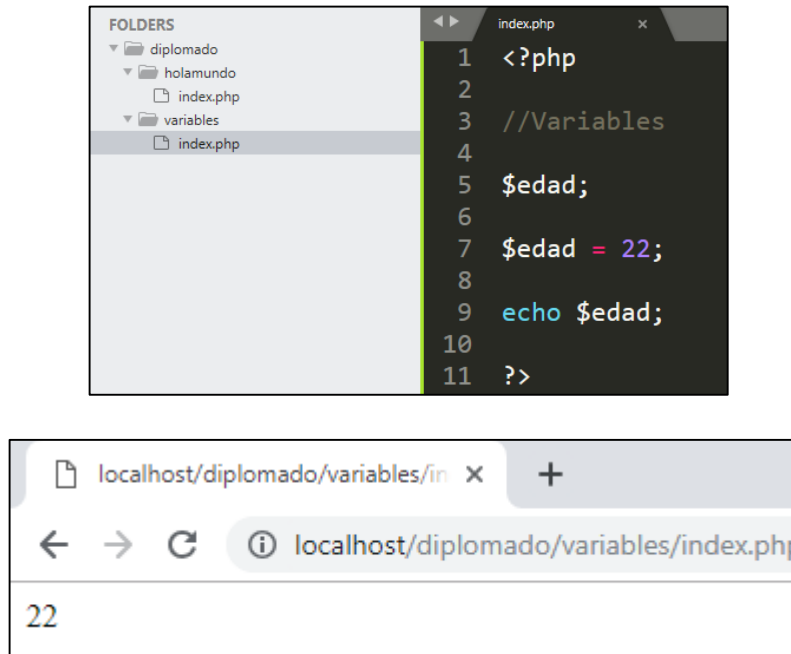
Ilustraciones 39 y 40.

Fuente: autor.

El error que representa la impresión de la variable edad, por medio de echo, se da textualmente porque la variable edad no ha sido declarada correctamente, no

cuenta con valor o no ha sido inicializada, y PHP muestra respectivamente ese error, e incluso indica la línea donde este se encuentra.

En el siguiente caso no habría problema en la declaración de edad:



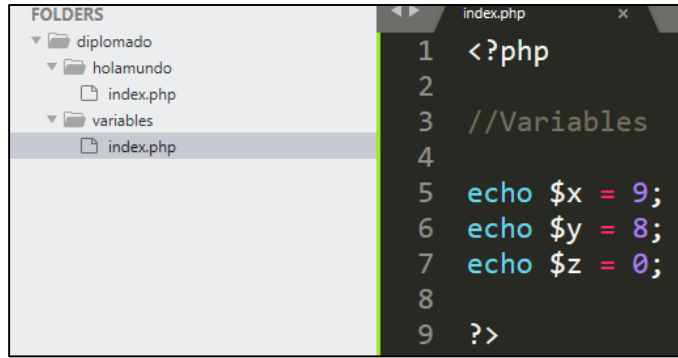
Ilustraciones 41 y 42.

Fuente: autor.

En este caso no hay ningún problema, se declara la variable en la línea 5, se asigna el valor de 22 en la línea 7 y finalmente se imprime el mismo valor en el navegador en la línea 9.

Nota: recuerda que para obtener los resultados en el navegador se debe volver a realizar la petición al mismo, o simplemente sobre el archivo o la ubicación de la URL en el navegador presionar F5. También, a su vez, tener en cuenta el guardar cada cambio que se realice en Sublime Text por medio de control + S.

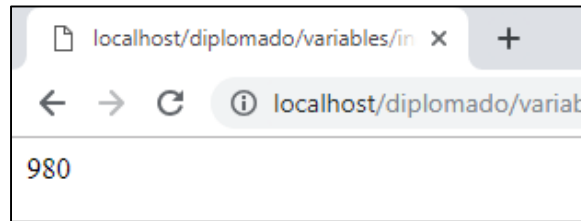
- ✓ Las variables pueden tener nombres cortos, como x, y, z.



```

1 <?php
2
3 //Variables
4
5 echo $x = 9;
6 echo $y = 8;
7 echo $z = 0;
8
9 ?>

```

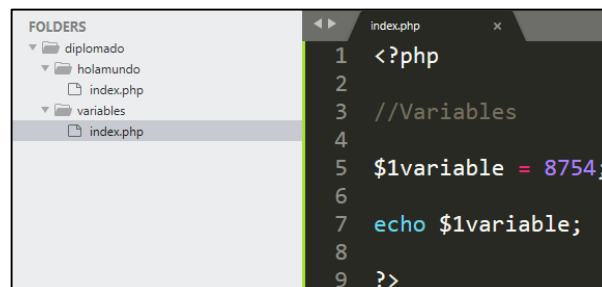


Ilustraciones 43 y 44.

Fuente: autor.

Nota: el resultado no es 980, es el resultado de imprimir las tres variables x, y, z, solo que el resultado se imprime en conjunto, dado que no estamos indicando su separación.

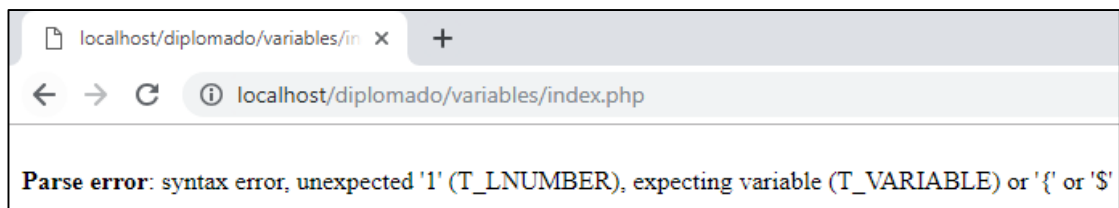
- ✓ Un nombre de variable no puede comenzar con un número.



```

1 <?php
2
3 //Variables
4
5 $1variable = 8754;
6
7 echo $1variable;
8
9 ?>

```

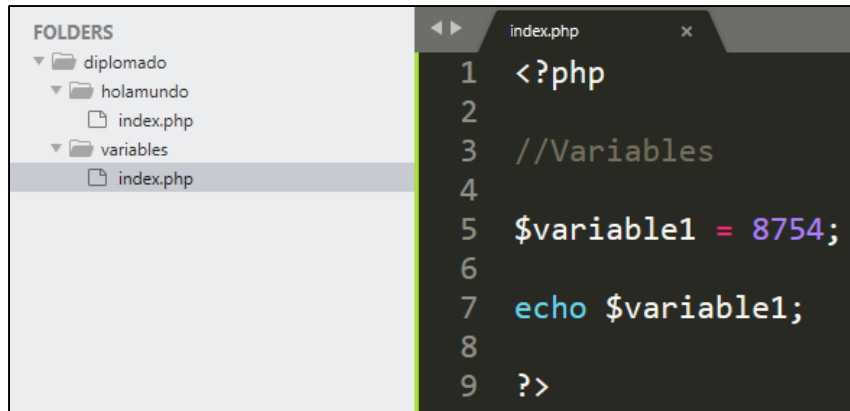


Ilustraciones 45 y 46: variables con número al inicio.

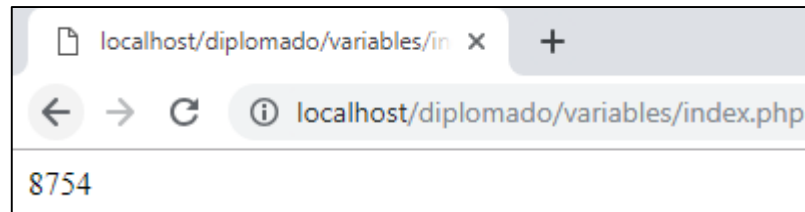
Fuente: autor.

El error es evidente, PHP espera el nombre de una variable y al toparse con un número (con previamente un símbolo de dólar) inmediatamente genera el error, evidenciado en la imagen anterior.

Lo que se puede hacer con los números en las variables es que estén presentes en otras posiciones menos la primera.



```
1 <?php
2
3 //Variables
4
5 $variable1 = 8754;
6
7 echo $variable1;
8
9 ?>
```

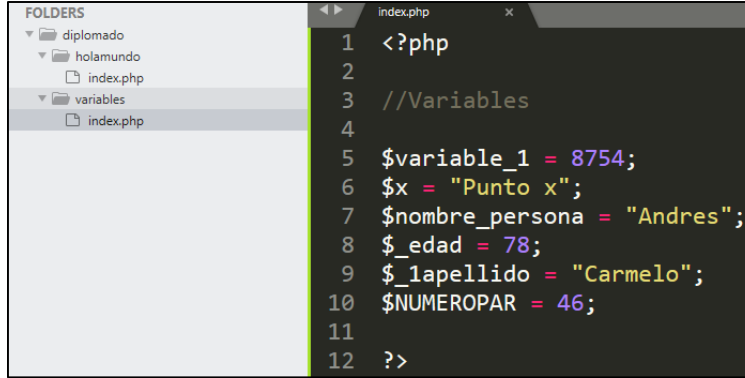


Ilustraciones 47 y 48.

Fuente: autor.

El resultado y la codificación es completamente válido, PHP permite sin problema alguno la inclusión de números en su declaración, siempre y cuando respeten la condición del primer carácter.

- ✓ Un nombre de variable solo puede contener caracteres alfanuméricos y guiones bajos (Az, 0-9 y _).

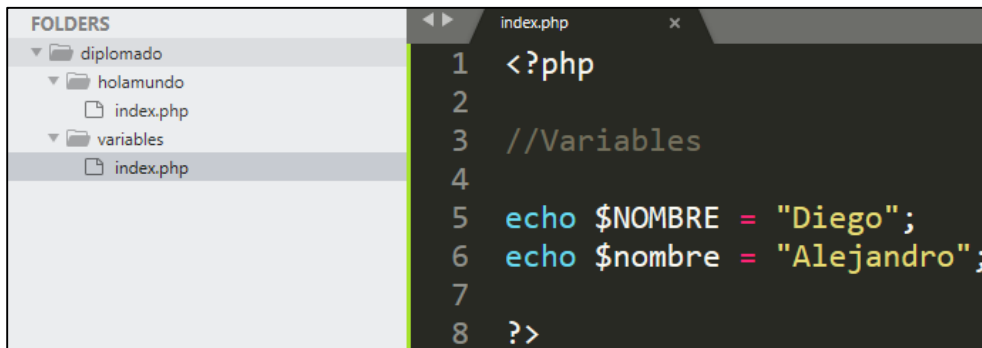


```
FOLDERS
  ▾ diplomado
    ▾ holamundo
      index.php
    ▾ variables
      index.php

index.php
1 <?php
2
3 //Variables
4
5 $variable_1 = 8754;
6 $x = "Punto x";
7 $nombre_persona = "Andres";
8 $_edad = 78;
9 $_lapellido = "Carmelo";
10 $NUMEROPAR = 46;
11
12 ?>
```

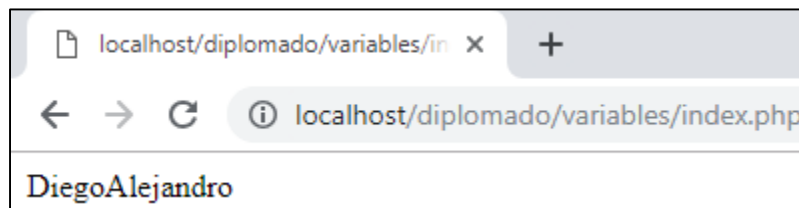
Ilustración 49.
Fuente: autor.

- ✓ Los nombres de las variables distinguen entre mayúsculas y minúsculas.



```
FOLDERS
  ▾ diplomado
    ▾ holamundo
      index.php
    ▾ variables
      index.php

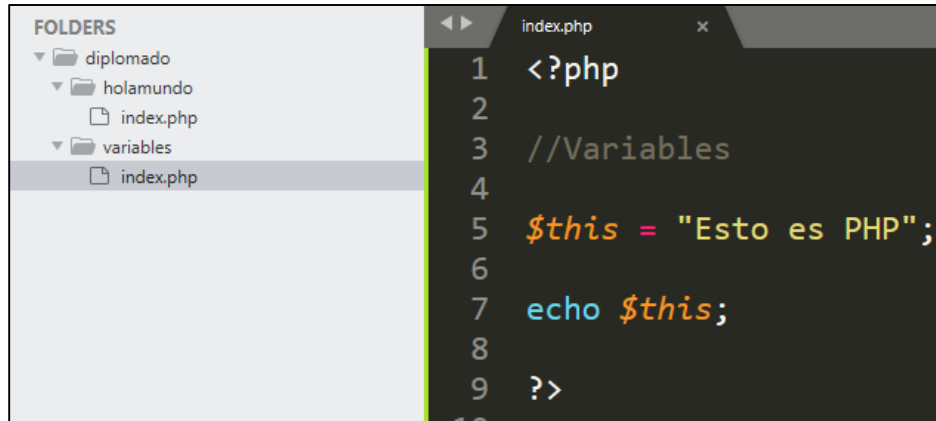
index.php
1 <?php
2
3 //Variables
4
5 echo $NOMBRE = "Diego";
6 echo $nombre = "Alejandro";
7
8 ?>
```



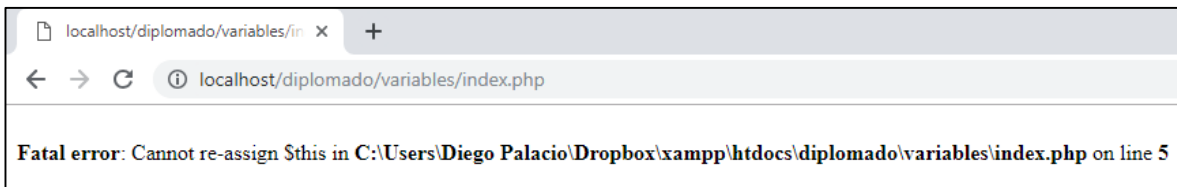
Ilustraciones 50 y 51.
Fuente: autor.

PHP diferencia completamente entre mayúsculas y minúsculas, no es lo mismo NOMBRE que nombre, pese a que sean «igual», textualmente hablando.

- ✓ This es una variable especial en PHP que no puede ser asignada.



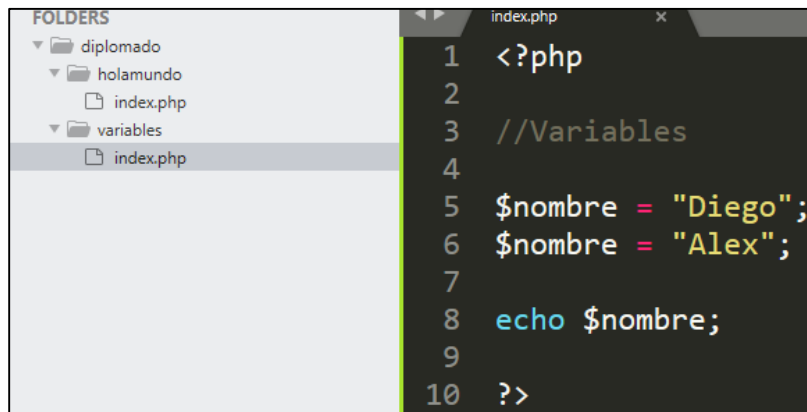
```
1 <?php
2
3 //Variables
4
5 $this = "Esto es PHP";
6
7 echo $this;
8
9 ?>
```



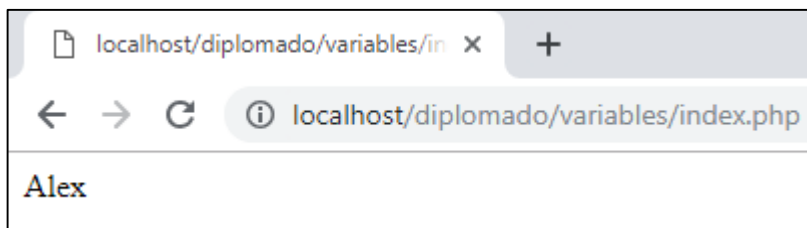
Ilustraciones 52 y 53.

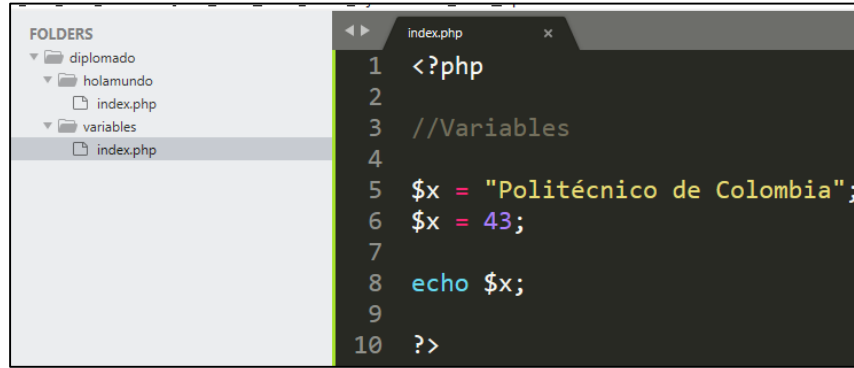
Fuente: autor.

- ✓ El valor de una variable puede ser reemplazado por otro, sin importar el tipo, claramente.

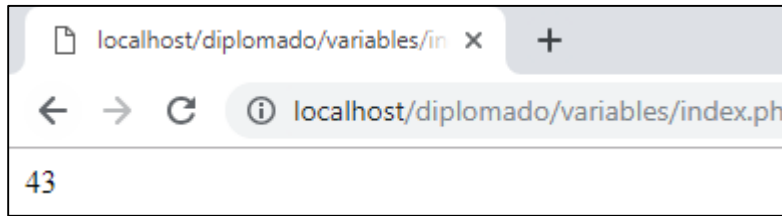


```
1 <?php
2
3 //Variables
4
5 $nombre = "Diego";
6 $nombre = "Alex";
7
8 echo $nombre;
9
10 ?>
```





```
1 <?php
2
3 //Variables
4
5 $x = "Politécnico de Colombia";
6 $x = 43;
7
8 echo $x;
9
10 ?>
```



Ilustraciones 54, 55, 56 y 57.

Fuente: autor.

Respectivamente el cambio de valores es evidente; al PHP, al no ser tipado, no hay problema en el intercambio de valores a partir de su contenido asignado en ambos casos, de «Diego» a «Alex» y de «Politécnico de Colombia» a «43». Cambios realizados sin ningún problema, teniendo en cuenta claramente que el valor de las variables reemplazadas («Diego» y «Politécnico de Colombia») se pierde al realizarse el intercambio.

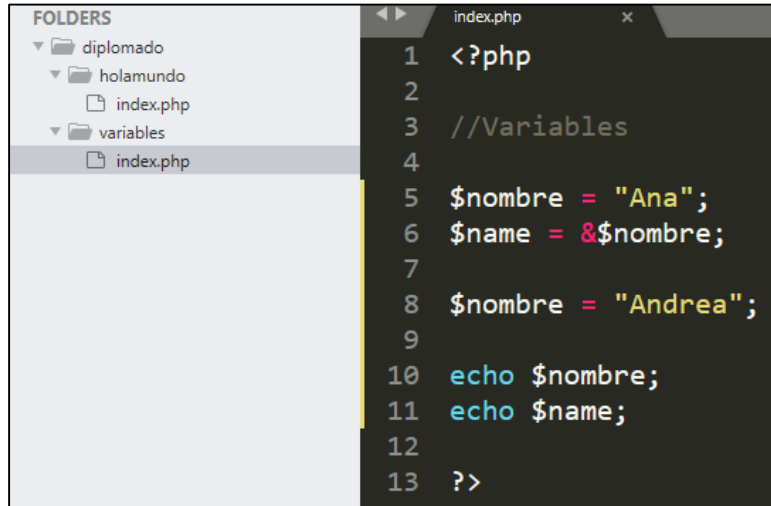
Por otra parte, de forma predeterminada, las variables siempre se asignan por valor. Esto significa que cuando se asigna una expresión a una variable el valor completo de la expresión original se copia en la variable de destino. Esto quiere decir que, por ejemplo, después de asignar el valor de una variable a otra, los cambios que se efectúen a una de esas variables no afectarán a la otra, caso contrario de las variables por referencia.

Variables por referencia

PHP también ofrece otra forma de asignar valores a las variables: asignar por referencia. Esto significa que la nueva variable simplemente referencia (en otras

palabras, «se convierte en un alias de» o «apunta a») la variable original. Los cambios a la nueva variable afectan a la original, y viceversa.

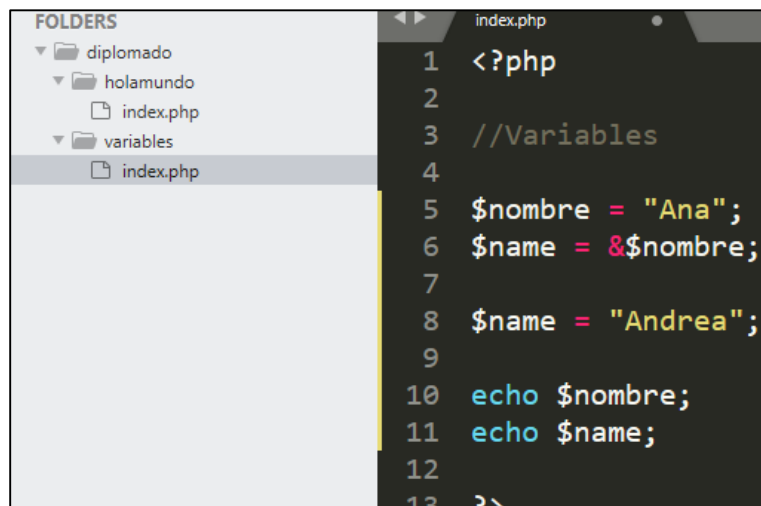
Para asignar por referencia se antepone un signo *ampersand* (&) al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo:



```

FOLDERS
  ▾ diplomado
    ▾ holamundo
      index.php
    ▾ variables
      index.php
  index.php

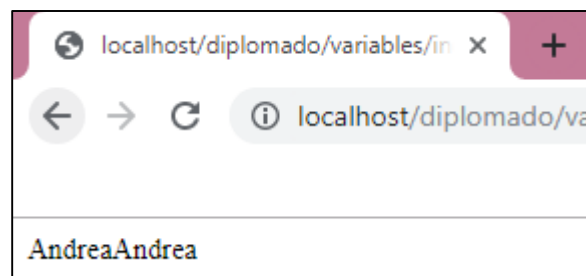
index.php
1  <?php
2
3  //Variables
4
5  $nombre = "Ana";
6  $name = &$nombre;
7
8  $nombre = "Andrea";
9
10 echo $nombre;
11 echo $name;
12
13 ?>
  
```



```

FOLDERS
  ▾ diplomado
    ▾ holamundo
      index.php
    ▾ variables
      index.php
  index.php

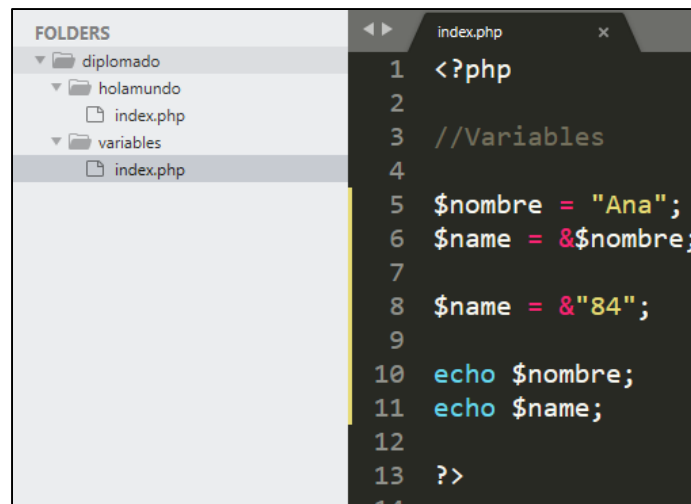
index.php
1  <?php
2
3  //Variables
4
5  $nombre = "Ana";
6  $name = $nombre;
7
8  $name = "Andrea";
9
10 echo $nombre;
11 echo $name;
12
13 ?>
  
```



Ilustraciones 58, 59 y 60.
Fuente: autor.

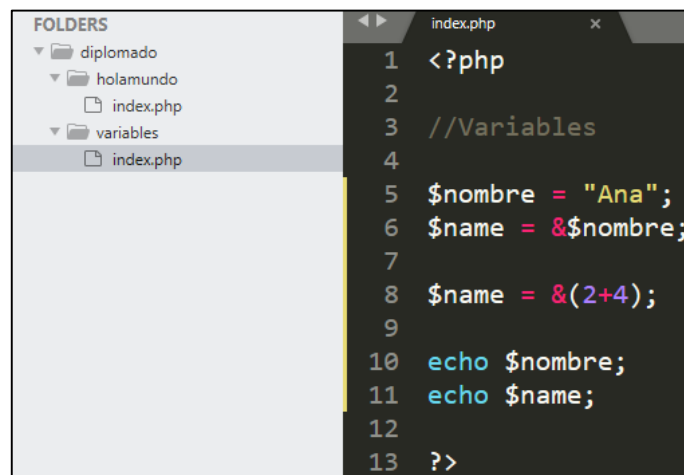
Lo que acaba de ocurrir con las dos variables descritas en la ilustración anterior es lo que se conoce como **variables por referencia**; en el ejemplo, a la variable *name* se le asigna una referencia de la variable *nombre*, al estar referenciada los cambios que le ocurran a una se verán reflejados en la otra y viceversa; por eso se presenta el mismo resultado en ambos casos, pese a que el cambio de «valor» se aplicó únicamente sobre *nombre*.

Una característica a tener en cuenta dentro de las variables por referencia es que únicamente se pueden referenciar variables que tengan un nombre, es decir, no se puede referenciar una operación o un valor, únicamente variables.



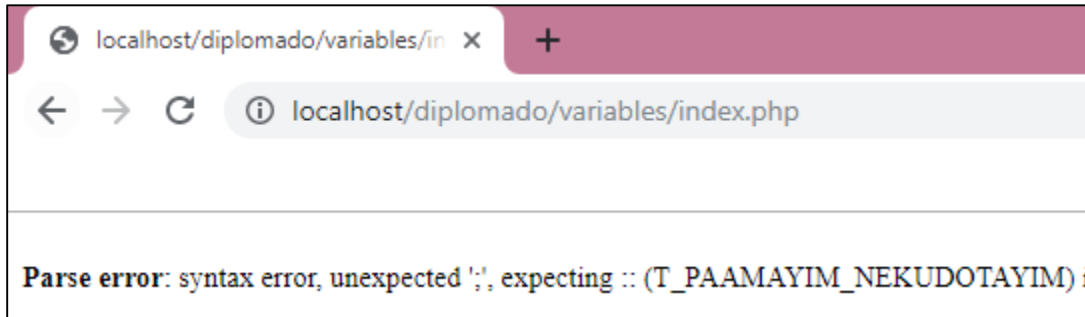
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder structure: 'diplomado' containing 'holamundo' and 'variables', each with an 'index.php' file. The code editor shows the following PHP code:

```
1 <?php
2
3 //Variables
4
5 $nombre = "Ana";
6 $name = &$nombre;
7
8 $name = &"84";
9
10 echo $nombre;
11 echo $name;
12
13 ?>
```



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the same folder structure as the previous screenshot. The code editor shows the following PHP code:

```
1 <?php
2
3 //Variables
4
5 $nombre = "Ana";
6 $name = &$nombre;
7
8 $name = &(2+4);
9
10 echo $nombre;
11 echo $name;
12
13 ?>
```



Ilustraciones 61, 62 y 63: variables por referencia con error.

Fuente: autor.

Los errores se presentan dado que PHP busca una variable para ser vinculada/referenciada con otra, y se encuentra con un valor/información/operación/función que no espera y genera el error.

Inicialización de variables y valores por defecto

No es necesario inicializar variables en PHP, sin embargo, es una muy buena práctica. Las variables no inicializadas tienen un valor predeterminado de acuerdo con su tipo dependiendo del contexto en el que son usadas —las booleanas se asumen como FALSE, los enteros y flotantes como cero, las cadenas se establecen como una cadena vacía y los *arrays* se convierten en un *array* vacío—.

Variables predefinidas

PHP proporciona gran cantidad de variables predefinidas que pueden ser ejecutadas en cualquier lugar y *script* de PHP, estas serán vistas más adelante en el diplomado. Algunas son:

- ✓ \$GLOBALS
- ✓ \$_SERVER
- ✓ \$_GET
- ✓ \$_POST
- ✓ \$_FILES
- ✓ \$_REQUEST
- ✓ \$_SESSION
- ✓ \$_ENV

✓ \$_COOKIE

Ámbito de las variables

En PHP las variables se pueden declarar en cualquier parte de los archivos y tienen un alcance, el alcance de una variable es la parte de la secuencia de comandos donde la variable puede ser referenciada/utilizada.

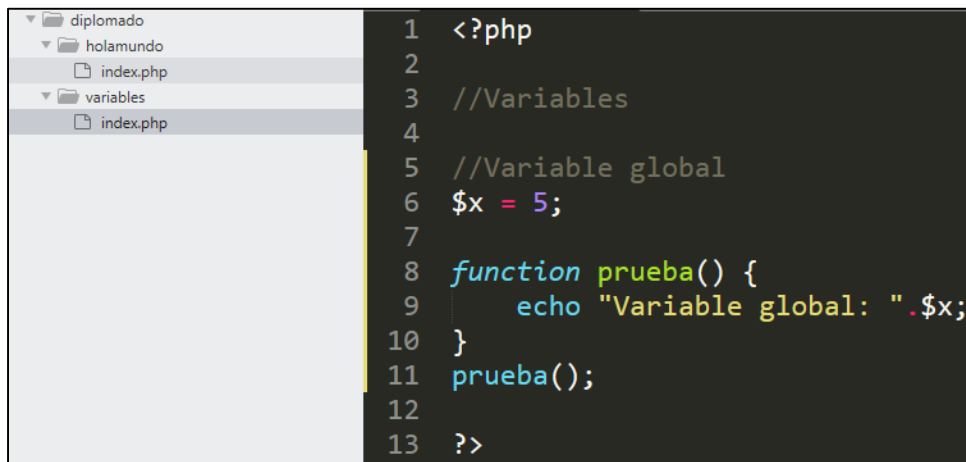
Existen tres ámbitos de variables diferentes:

- ✓ Local
- ✓ Global
- ✓ Estático

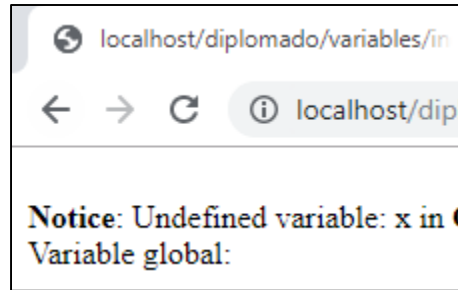
Para entender el funcionamiento y aplicación de los ámbitos, entrarán en juego algunos nuevos conceptos; no te preocupes, estos serán ampliados fuertemente en el transcurso de las guías didácticas.

Ámbito global - Ámbito local

Una variable declarada fuera de una función tiene un ámbito global y no puede ser accedida desde una función (ámbito local):



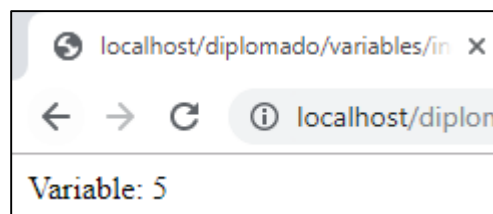
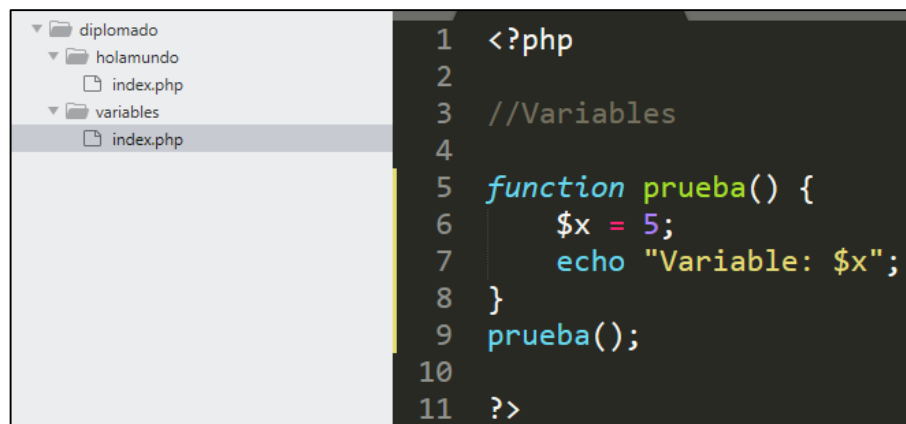
```
1 <?php
2
3 //Variables
4
5 //Variable global
6 $x = 5;
7
8 function prueba() {
9     echo "Variable global: ".$x;
10 }
11 prueba();
12
13 ?>
```

Ilustraciones 64 y 65.
Fuente: autor.

Las funciones (*function*) son rutinas de código que pueden ser llamadas/solicitadas una o más veces en cualquier momento para repetir un proceso, en este caso la función *prueba* tiene encargado el proceso de imprimir el valor de la variable *x* anteriormente declarada y que por su contexto es global. Esta instrucción/función generará error, dado que las variables de ámbito global no pueden ser accedidas en un ámbito local, en este caso el ámbito local es la función *prueba*.

En caso de querer obtener esa variable en el código de la función, lo que se debe hacer es declararla dentro de la función para que sea una variable de ámbito local y acceder a su valor sin ningún problema:



Ilustraciones 66 y 67.

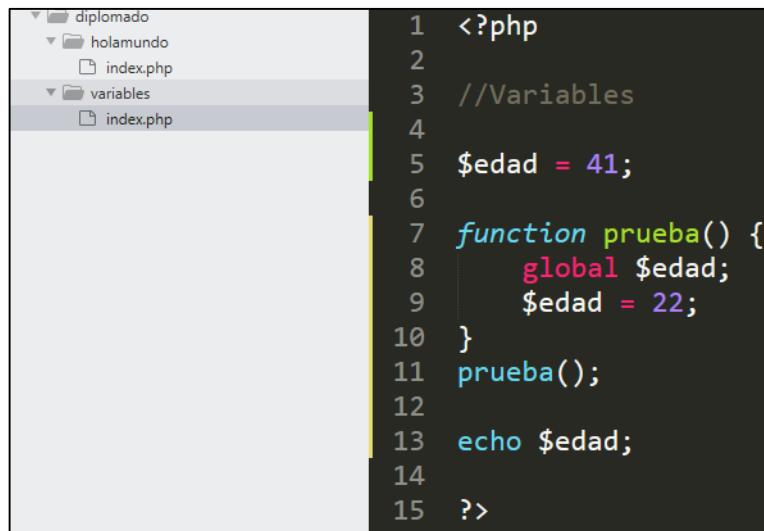
Fuente: autor.

Un aspecto para tener en cuenta es que se pueden tener variables locales con el mismo nombre en diferentes funciones, porque las variables locales solo son reconocidas por la función en la que están declaradas.

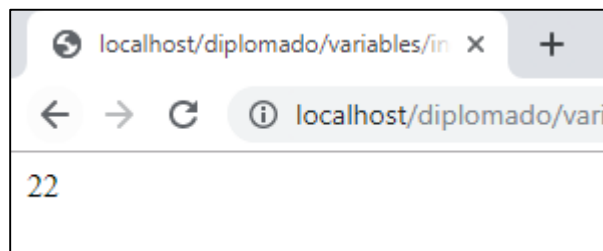
Variable global

La palabra reservada **global** se utiliza para acceder a una variable global desde dentro de una función.

Para esto, se debe usar la palabra reservada antes de las variables (dentro de la función):



```
1 <?php
2
3 //Variables
4
5 $edad = 41;
6
7 function prueba() {
8     global $edad;
9     $edad = 22;
10 }
11 prueba();
12
13 echo $edad;
14
15 ?>
```



Ilustraciones 68 y 69.

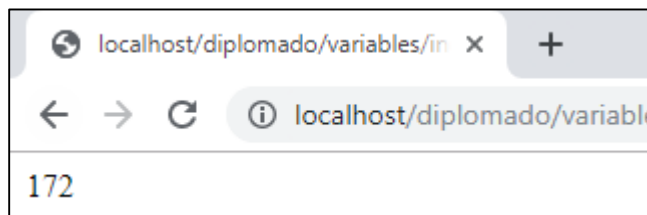
Fuente: autor.

Este proceso puede verse como redeclaración de la variable de ámbito global ahora en un ámbito local, haciendo referencia siempre a la variable global.

Otra forma de obtener este resultado es haciendo uso de una de las variables predefinidas de PHP: **\$GLOBALS**.

\$GLOBALS representa el grupo de variables globales declaradas, esta variable guarda el nombre textual de todas las variables anteriormente declaradas y, gracias a que almacena el nombre, los valores pueden ser leídos y sobrescritos con facilidad.

```
1  <?php
2
3  //Variables
4
5  $altura = 141;
6
7  function prueba()
8  {
9      $GLOBALS['altura'] = 172;
10 }
11
12 prueba();
13
14 echo $altura;
15
16 ?>
```



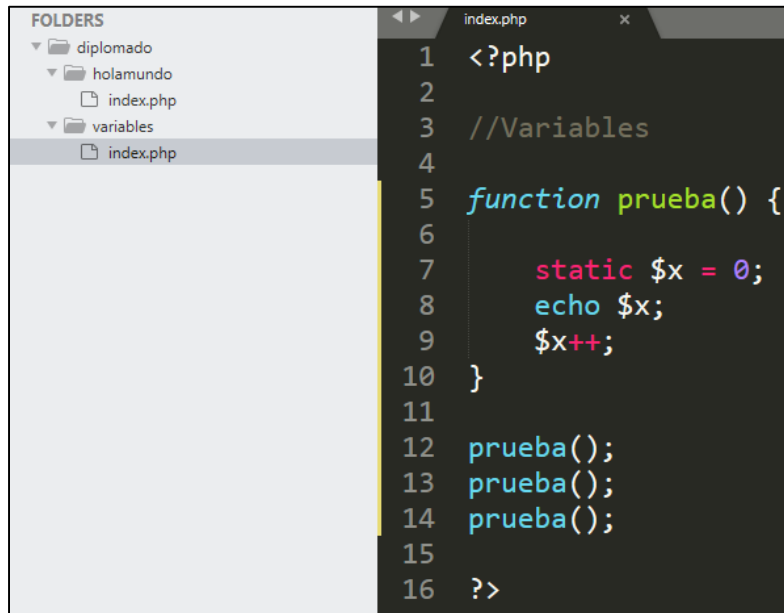
Ilustraciones 70 y 71.

Fuente: autor.

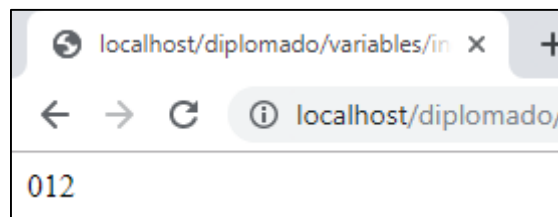
Gracias a **\$GLOBALS** se pueden obtener los mismos resultados al igual que global o declarando la variable de ámbito local. Son diferentes formas de llegar a una pequeña solución (W3Schools, 2019).

Variables estáticas

Las variables estáticas deben su nombre a que son variables que cuentan con la palabra clave **static**. En el ámbito de las variables, *static* brinda a las variables locales la posibilidad de retener información en tiempo de ejecución, es decir, en el llamado a una función todos los valores contenidos en estas se reinician cada vez que se realiza un llamado diferente, con *static*, alguna propiedad/variable puede retener información durante todo el tiempo de ejecución, para esto véase el siguiente ejemplo:



```
1 <?php
2
3 //Variables
4
5 function prueba() {
6
7     static $x = 0;
8     echo $x;
9     $x++;
10 }
11
12 prueba();
13 prueba();
14 prueba();
15
16 ?>
```



Ilustraciones 72 y 73.

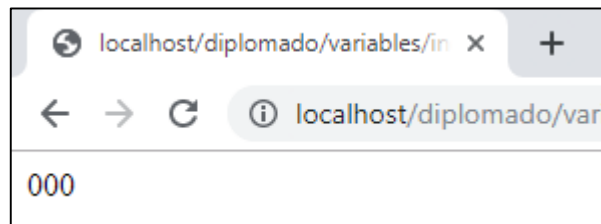
Fuente: autor.

En este caso la variable `$x`, al contener la propiedad **static**, está reteniendo información en tiempo de ejecución. El proceso que realiza la función se basa en crear una variable `$x`, imprimir su valor y posteriormente incrementar este a razón de 1 (`++`), este proceso de incremento será explicado posteriormente, en caso de que la propiedad *static* no estuviese presente, el resultado de impresión del `echo`

sería 0 en tres ocasiones, porque en cada llamado a la función esta reiniciaría el valor de \$x. Véase:



```
1 <?php
2
3 //Variables
4
5 function prueba() {
6
7     $x = 0;
8     echo $x;
9     $x++;
10 }
11
12 prueba();
13 prueba();
14 prueba();
15
16 ?>
```



Ilustraciones 74 y 75.

Fuente: autor.

Variables constantes

Las constantes son elementos de PHP que guardan un valor fijo que no se puede modificar a lo largo del programa. Las constantes pueden ser definidas por el programa o estar predefinidas por el propio PHP o por algún módulo. Los nombres de las constantes siguen las mismas reglas que los nombres de las variables, pero sin el símbolo (\$) inicial. La costumbre es escribir los nombres de las constantes en mayúsculas.

Existen dos formas para declarar variables constantes en PHP:

- ✓ *Const*
- ✓ *Define*

La primera forma, haciendo uso de la *const*:

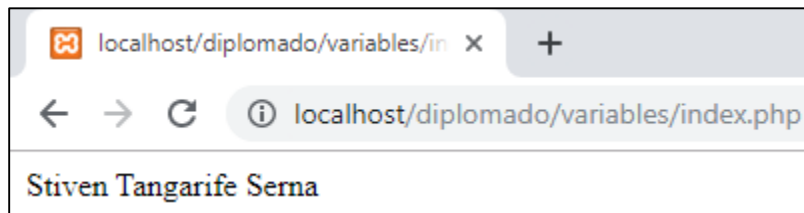
```
<?php

//Variables Constantes

const ESTUDIANTE = "Stiven Tangarife Serna";

echo ESTUDIANTE;

?>
```



Ilustraciones 76 y 77.

Fuente: autor.

La segunda forma, haciendo uso sobre *define*:

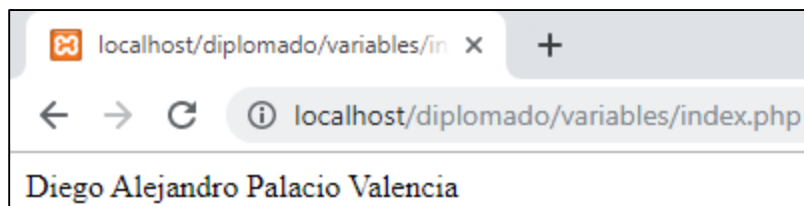
```
<?php

//Variables Constantes

define("PROFESOR", "Diego Alejandro Palacio Valencia");

echo PROFESOR;

?>
```



Ilustraciones 78 y 79.

Fuente: autor.

En principio, se puede no utilizar constantes nunca, puesto que las constantes definidas por el programa podrían reemplazarse por variables. La ventaja de usar constantes y variables es que se puede distinguir a simple vista si a lo largo de un programa algo va a permanecer constante (si es una constante) o puede cambiar (si es una variable).

La particularidad de las constantes es la imposibilidad de cambiar los valores de estas:

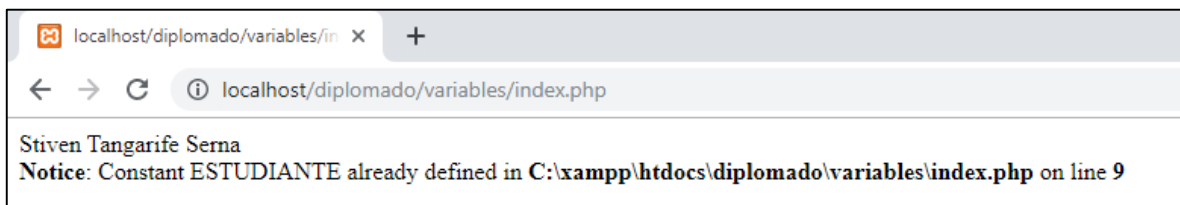
```
<?php
//Variables Constantes

const ESTUDIANTE = "Stiven Tangarife Serna";

echo ESTUDIANTE;

const ESTUDIANTE = "Juan Rivera";

?>
```



Ilustraciones 80 y 81.

Fuente: autor.

Ejercicios con variables

Para comprender mejor el funcionamiento y operación de las variables, se presentan algunos ejercicios prácticos:

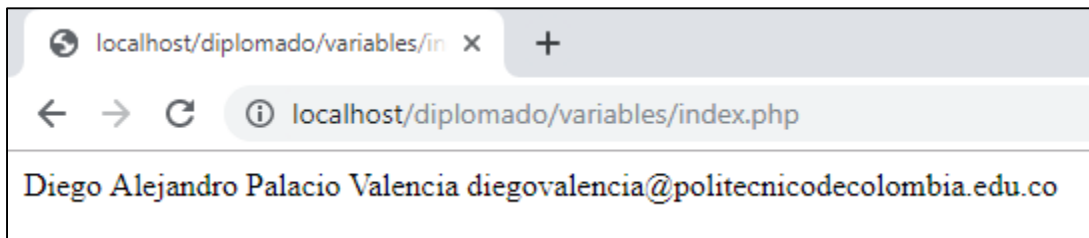
1. Crear dos variables, una para almacenar el nombre completo de una persona y otra para almacenar el correo electrónico e imprimir las dos variables en un *echo*.

Forma 1:


```
index.php x
1 <?php
2
3 //Variables
4
5 $nombre = "Diego Alejandro Palacio Valencia";
6 $correo = "diegovalencia@politecnicodecolombia.edu.co";
7
8 echo $nombre." ".$correo;
9
10 ?>
```

Forma 2:

```
index.php x
1 <?php
2
3 //Variables
4
5 $nombre = "Diego Alejandro Palacio Valencia";
6 $correo = "diegovalencia@politecnicodecolombia.edu.co";
7
8 echo "$nombre $correo";
9
10 ?>
```



Ilustraciones 82, 83 y 84.

Fuente: autor.

Hay dos formas de llegar a la solución, se puede decir que lo más complicado es imprimir dos variables en un `echo`, hasta el momento en la mayoría de los casos en los que se hizo uso del `echo`, se hacía con solo una variable o un dato, de estas formas:

```
index.php x
1  <?php
2
3  //Variables
4
5  $nombre = "Diego Alejandro Palacio Valencia";
6  $correo = "diegovalencia@politecnicodecolombia.edu.co";
7
8  echo $nombre;
9  echo "Hola Mundo";
10 echo "El correo es: $correo";
11 echo "El correo es: ".$correo;
12 echo "$nombre";
13
14 ?>
```

Ilustración 85.

Fuente: autor.

Pero PHP, y más directamente *echo*, permite imprimir X cantidad de variables en una sola declaración del *echo*, pueden realizarse todas dentro de las comillas al igual que en la Forma 1, conservando los espacios entre variables, o concatenando al igual que en la Forma 2. Concatenar es un concepto nuevo, esta concatenación permite unir dos variables por medio del punto (.), estos son algunos de los ejemplos de la concatenación en conjunto con las comillas a forma de separación (un espacio entre cada variable para entender mejor el contenido de cada una).

```
index.php x
1  <?php
2
3  //Variables
4
5  $nombre = "Diego Alejandro Palacio Valencia";
6  $correo = "diegovalencia@politecnicodecolombia.edu.co";
7
8  echo $nombre.$correo;
9  echo $nombre." ".$correo;
10 echo "Nombre: ".$nombre." Correo: ".$correo;
11
12 ?>
```

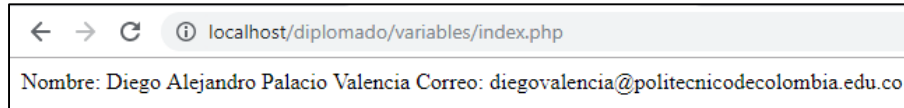
Ilustración 86.

Fuente: autor.

Esto facilita la impresión de resultados, dado que ya no se haría uso de un *echo* por variable o información a imprimir, sino que permite en una sola declaración

realizar este trabajo. Para convención del diplomado, en mayor medida se utilizará la impresión en PHP de la siguiente forma para ser más claro y legible el código para la correcta comprensión, aunque es válido usar la forma que se considere correcta y práctica por parte del estudiante:

```
echo "Nombre: ".$nombre." Correo: ".$correo;
```



Ilustraciones 87 y 88.

Fuente: autor.

Crear tres variables para tres números, de 0 a 9, cada uno diferente del otro, es decir, sin repetir; imprimir todas las combinaciones posibles entre los números para formar números de tres dígitos. Ejemplo:

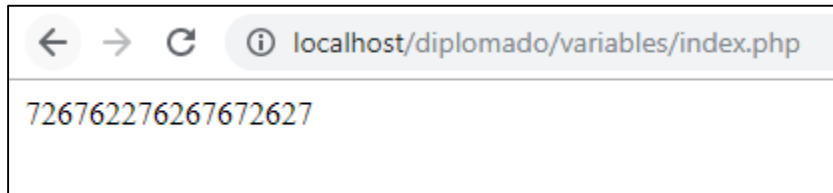
Var1 = 8;

Var2 = 3;

Var3 = 5;

835 - 538 - 583 - 385 - 358 - 853

```
index.php x
1  <?php
2
3  //Variables
4
5  $numero1 = 7;
6  $numero2 = 2;
7  $numero3 = 6;
8
9  echo $numero1.$numero2.$numero3;
10 echo $numero1.$numero3.$numero2;
11 echo $numero2.$numero1.$numero3;
12 echo $numero2.$numero3.$numero1;
13 echo $numero3.$numero1.$numero2;
14 echo $numero3.$numero2.$numero1;
15
16 ?>
```



Ilustraciones 89 y 90.

Fuente: autor.

Esta es la solución para el ejercicio, aunque el resultado es un poco complicado para visualizar, se puede mejorar de dos formas:

- ✓ Imprimiendo un guion por cada tres dígitos.
- ✓ Imprimiendo en un renglón diferente cada tres dígitos.

Véase las dos posibles soluciones, se hará uso de la primera etiqueta de HTML para realizar los saltos de línea o renglón.

1. Guion por cada tres dígitos:

```
index.php x
1 <?php
2
3 //Variables
4
5 $numero1 = 7;
6 $numero2 = 2;
7 $numero3 = 6;
8
9 echo $numero1.$numero2.$numero3." - ";
10 echo $numero1.$numero3.$numero2." - ";
11 echo $numero2.$numero1.$numero3." - ";
12 echo $numero2.$numero3.$numero1." - ";
13 echo $numero3.$numero1.$numero2." - ";
14 echo $numero3.$numero2.$numero1;
15
16 ?>
```

localhost/diplomado/variables/index.php
726 - 762 - 276 - 267 - 672 - 627

Ilustraciones 91 y 92.

Fuente: autor.

Esta solución es completamente válida, se hace la respectiva concatenación entre las tres variables y adicionalmente concatena la cadena texto que contiene el guion que los separa. Este mismo resultado podría obtenerse en un solo *echo* sin ningún problema, solo que la declaración sería un poco extensa; con este ejemplo queda un poco más claro que *echo* permitirá concatenar las variables que se desees dentro de su declaración.

2. Renglón diferente para cada tres dígitos.

```
index.php x
1 <?php
2
3 //Variables
4
5 $numero1 = 7;
6 $numero2 = 2;
7 $numero3 = 6;
8
9 echo $numero1.$numero2.$numero3."<br>";
10 echo $numero1.$numero3.$numero2."<br>";
11 echo $numero2.$numero1.$numero3."<br>";
12 echo $numero2.$numero3.$numero1."<br>";
13 echo $numero3.$numero1.$numero2."<br>";
14 echo $numero3.$numero2.$numero1;
15
16 ?>
```

localhost/diplomado/variables/index.php

726
762
276
267
672
627

Ilustraciones 93 y 94.

Fuente: autor.

 es la primera etiqueta de código HTML, esta instrucción, básicamente, se encarga de realizar un salto de línea donde esté ubicada; en este caso, al final de cada `echo` realiza el salto de línea al próximo conjunto de tres números, aquí otro ejemplo del funcionamiento de
. A lo largo del diplomado poco a poco se irán conociendo las etiquetas de HTML.

```
index.php x
1 Aquí estamos escribiendo <br> texto en SublimeText
```

localhost/diplomado/variables/index.php

Aquí estamos escribiendo
texto en SublimeText

Ilustraciones 95 y 96.

Fuente: autor.

Corrija/organice todas las siguientes declaraciones a partir de lo aprendido sobre las variables en PHP, el uso de *echo* y las funciones, de manera tal que no se presenten errores en el navegador.

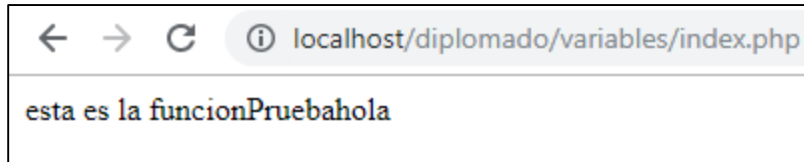
```
index.php x
1  <?php
2
3  //Variables
4
5  funcionPrueba();
6
7  altura = 1.72;
8  $x = 7;
9  $nombre = Diego;
10 $_GLOBALS['x'] = 8;
11 $pais_persona = "Colombia";
12 $pais = "Mi pais es"$pais_persona;
13 $valor = $this;
14 echo "hola"
15 $correo;
16 static $y = .04;
17 $ciudad "medellin";
18
19 function funcionPrueba()
20 {
21     eco "esta es la funcionPrueba";
22 }
23
24 ?>
```

Ilustración 97.

Fuente: autor.

Hay errores muy simples y otros un poco más lógicos, desde declaraciones incompletas, instrucciones sin cierre de punto y coma hasta igualdades de valores (=), esta es una solución aplicable al problema (The PHP Group, 2019).

```
index.php x
1  <?php
2
3  //Variables
4
5  function funcionPrueba()
6  {
7      echo "esta es la funcionPrueba";
8  }
9
10 funcionPrueba();
11
12 $altura = 1.72;
13 $x = 7;
14 $nombre = "Diego";
15 $GLOBALS['x'] = 8;
16 $pais_persona = "Colombia";
17 $pais = "Mi pais es ".$pais_persona;
18 $valor = 4542;
19 echo "hola";
20 $correo = "diegovalencia@politecnicodecolombia.edu.co";
21 static $y = 0.04;
22 $ciudad = "medellin";
23
24 ?>
```



← → ↻ ⓘ localhost/diplomado/variables/index.php

esta es la funcionPruebahola

Ilustraciones 98 y 99.

Fuente: autor.

Tema 5: Tipos de Datos

PHP es un lenguaje no tipado. Esto significa que las variables necesitan ser inicializadas y su tipo de dato no solo no precisa ser indicado, sino que este puede cambiar. PHP sabe el tipo de dato que se utiliza en cada momento dependiendo del contexto en que se utilice.

Según esto, hay formas de determinar el valor de una variable en PHP, pese a que no esté especificado en su declaración, pero sí por su valor.

PHP soporta los siguientes tipos de datos en su construcción:

- *String* (cadenas de texto)
- *Integer* (número enteros)
- *Float* (decimales)
- *Boolean*
- *Array*
- *Object*
- *Null*

Para explicar en mayor medida estos tipos de datos, y la forma de descifrar el tipo que contiene, se harán uso de nuevos conceptos que se explicarán en desarrollo del módulo y otros que serán tratados más adelante en el diplomado; así que sí denotan ser complejos, pero poco a poco se irá comprendiendo en mayor medida su operatividad.

Para continuar con el orden de codificación establecido para el diplomado, se debe crear un nuevo directorio o proyecto para contener el nuevo tema a trabajar, los tipos de datos, de la siguiente forma:

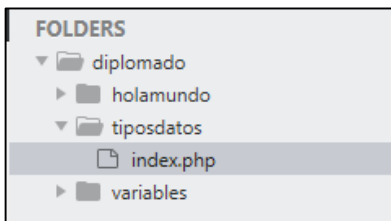
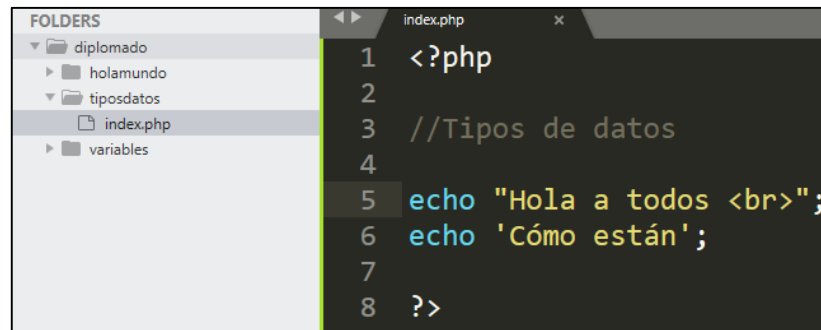


Ilustración 100.

Fuente: autor.

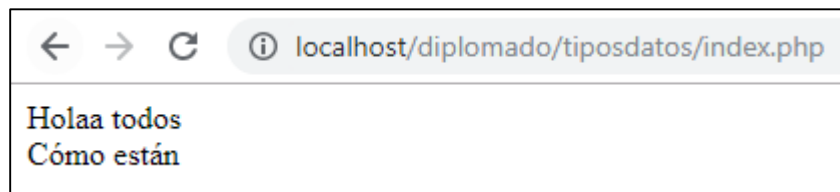
String (cadenas de texto)

Una cadena es una secuencia de caracteres, como «¡Hola mundo!». Una cadena puede ser cualquier texto entre comillas. Puedes usar comillas simples o dobles:



```
FOLDERS
└─ diplomado
  └─ tiposdatos
    └─ index.php
  └─ variables

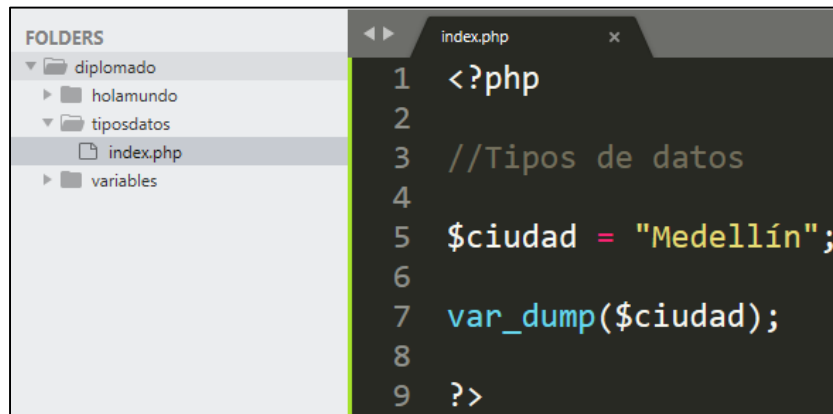
index.php
1 <?php
2
3 //Tipos de datos
4
5 echo "Hola a todos <br>";
6 echo 'Cómo están';
7
8 ?>
```



Ilustraciones 101 y 102.

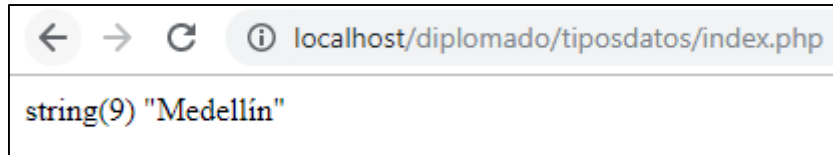
Fuente: autor.

La forma en la que PHP determina que es cadena de texto, *string* o cualquier otro tipo de dato es por medio de **var_dump**, esta función permite mostrar la información estructurada de una variable, incluyendo su tipo, valor y longitud. Un ejemplo más claro de esto será el siguiente:



```
FOLDERS
└─ diplomado
  └─ tiposdatos
    └─ index.php
  └─ variables

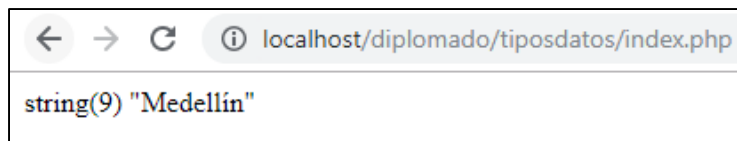
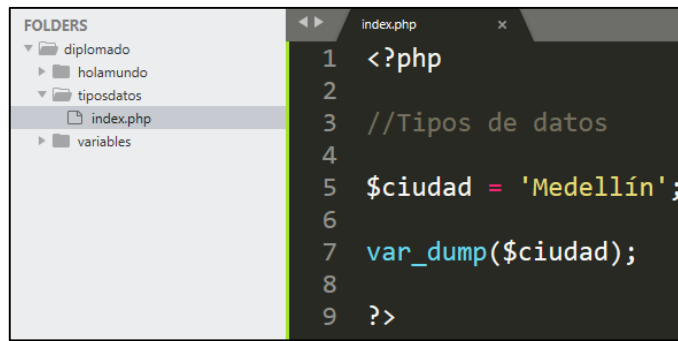
index.php
1 <?php
2
3 //Tipos de datos
4
5 $ciudad = "Medellín";
6
7 var_dump($ciudad);
8
9 ?>
```



Ilustraciones 103 y 104.

Fuente: autor.

El resultado, como se espera en la definición de `var_dump`, es tal cual, este retornará el tipo de dato que tiene asignada la variable en ese momento, la longitud de esta y por supuesto el valor. Lo mismo ocurre con las cadenas de texto declaradas con comillas simples:



Ilustraciones 105 y 106.

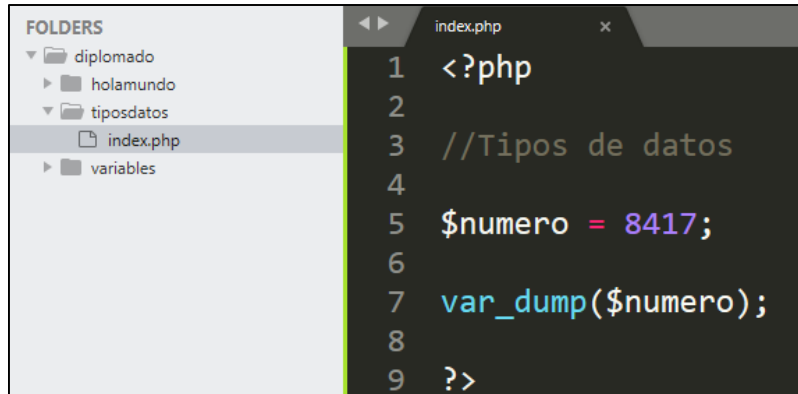
Fuente: autor.

Es simple pero funcional el uso de `var_dump` en relación con las variables, así mismo ocurre con los demás tipos.

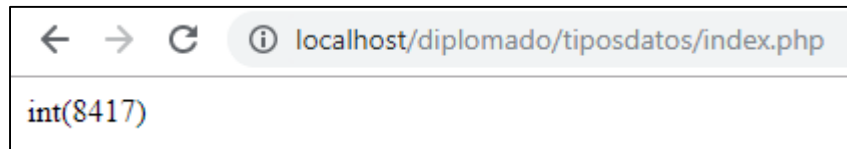
***Int / Integer* (números enteros)**

Un tipo de datos *Integer* o entero es un número no decimal entre -2147483648 y 2147483647.

Para observar y validar que efectivamente se trata de un número entero, hágase uso de `var_dump` sobre una variable que contenga un número entero:



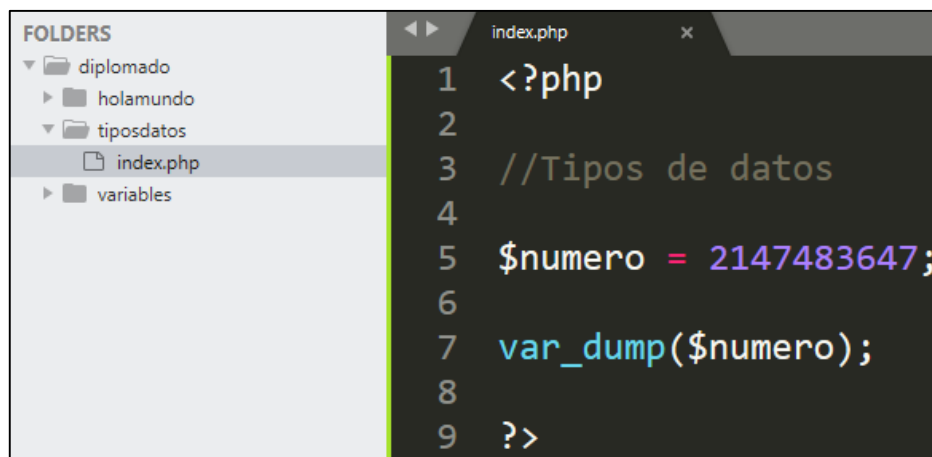
```
1 <?php
2
3 //Tipos de datos
4
5 $numero = 8417;
6
7 var_dump($numero);
8
9 ?>
```



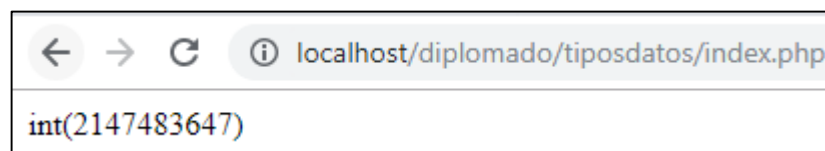
Ilustraciones 107 y 108.
Fuente: autor.

Para conocer la estructura de una variable de tipo entero no existe un tamaño, el tamaño está dictado por el valor que contenga la variable, así que solo se obtendrá la información del tipo y el valor.

Una forma de validar hasta qué rango un número es entero y pasa a ser un número decimal, por su longitud, es de la siguiente forma:



```
1 <?php
2
3 //Tipos de datos
4
5 $numero = 2147483647;
6
7 var_dump($numero);
8
9 ?>
```



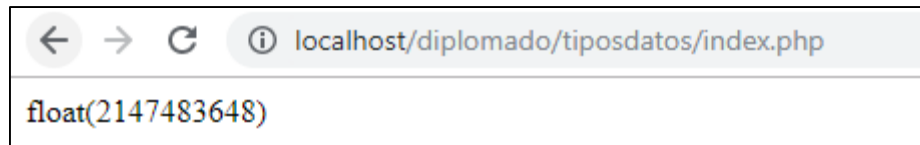
Ilustraciones 109 y 110.

Fuente: autor.

Este es el número máximo contenido en los números enteros, pasando de 2147483647 a 2147483648 el valor y el tipo cambian completamente:



```
1 <?php
2
3 //Tipos de datos
4
5 $numero = 2147483648;
6
7 var_dump($numero);
8
9 ?>
```



Ilustraciones 111 y 112.

Fuente: autor.

Ahora el valor y el tipo de la variable pertenecen a los números flotantes.

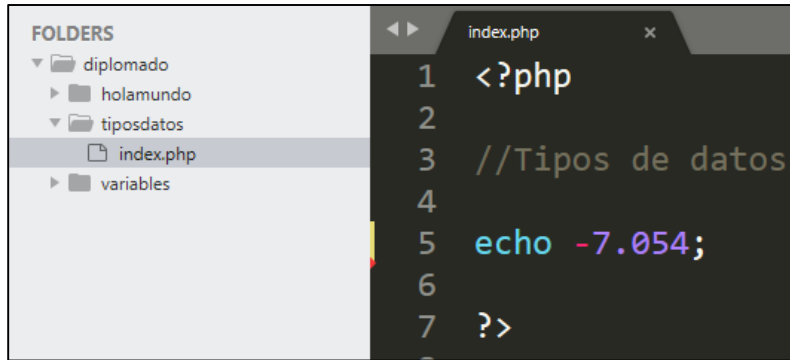
Float / Double (números decimales)

El tipo *float* o *double* (número de punto flotante) es un número con un punto decimal.

Hay características para tener en cuenta con el trabajo de números decimales:

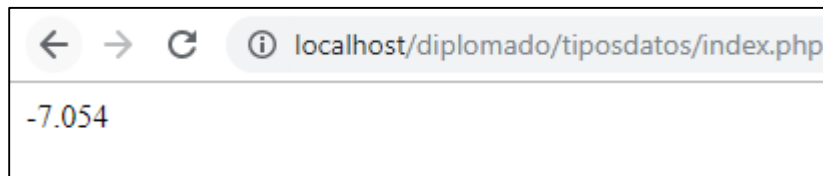
- ✓ Un *float* debe tener al menos un dígito decimal.
- ✓ Un *float* puede ser positivo o negativo.
- ✓ Un *float* no debe ir entre comillas simples o dobles.

Para entender de mejor forma cómo operan las variables *float* o los tipos de datos, obsérvese el siguiente ejemplo:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder structure: diplomado > holamundo > tiposdatos > index.php. The code editor shows the content of index.php:

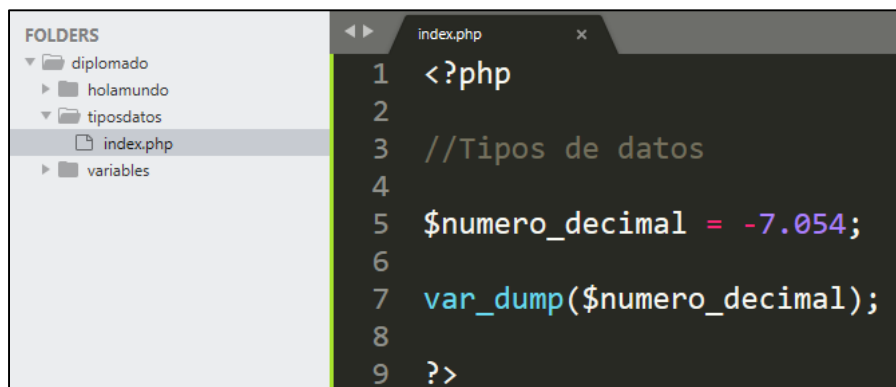
```
1 <?php
2
3 //Tipos de datos
4
5 echo -7.054;
6
7 ?>
```



Ilustraciones 113 y 114.

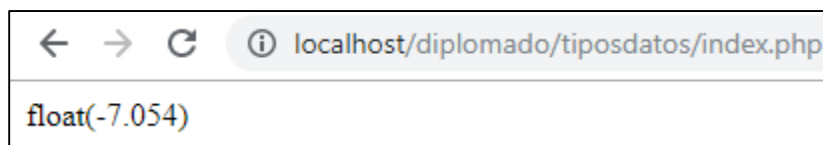
Fuente: autor.

Para observar y validar que efectivamente se trata de un número decimal, hágase uso de `var_dump` sobre un variable que contenga un número decimal, al igual que en los tipos de datos anteriores:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder structure: diplomado > holamundo > tiposdatos > index.php. The code editor shows the content of index.php:

```
1 <?php
2
3 //Tipos de datos
4
5 $numero_decimal = -7.054;
6
7 var_dump($numero_decimal);
8
9 ?>
```



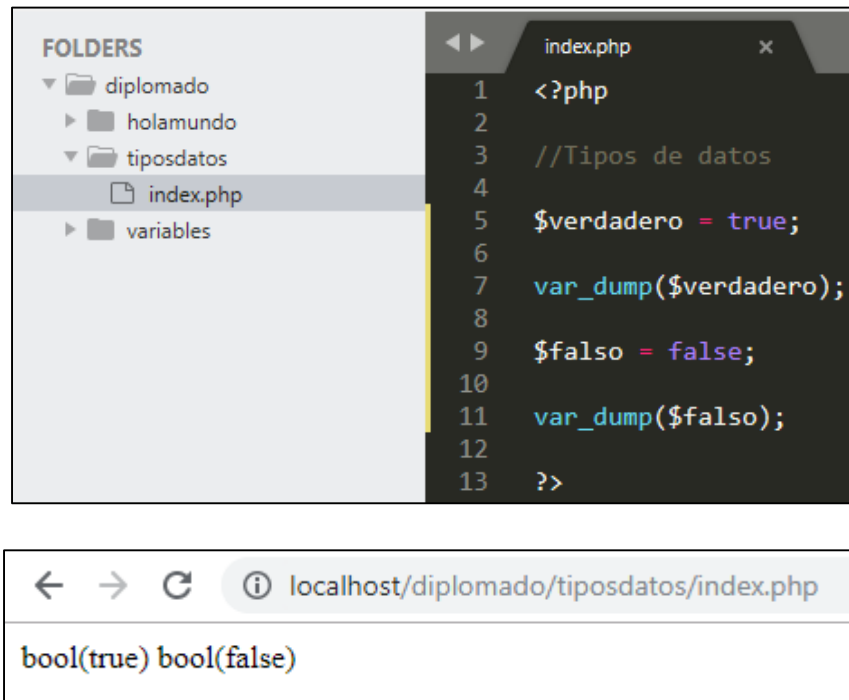
Ilustraciones 115 y 116.

Fuente: autor.

Al igual que ocurre con los tipos de datos enteros y variables, para conocer la estructura de una variable de tipo decimal no existe un tamaño para la variable, el tamaño está dictado por el valor que contenga la variable, así que solo se obtendrá la información del tipo y el valor.

Boolean (verdadero o falso)

El tipo de dato *boolean* solo representa dos posibles estados o valores: verdadero o falso (*true* o *false*).



Ilustraciones 117 y 118.

Fuente: autor.

Los *booleanos* se utilizan a menudo en las pruebas de condicionales. Se aprenderá más sobre las pruebas condicionales en un próximo tema.

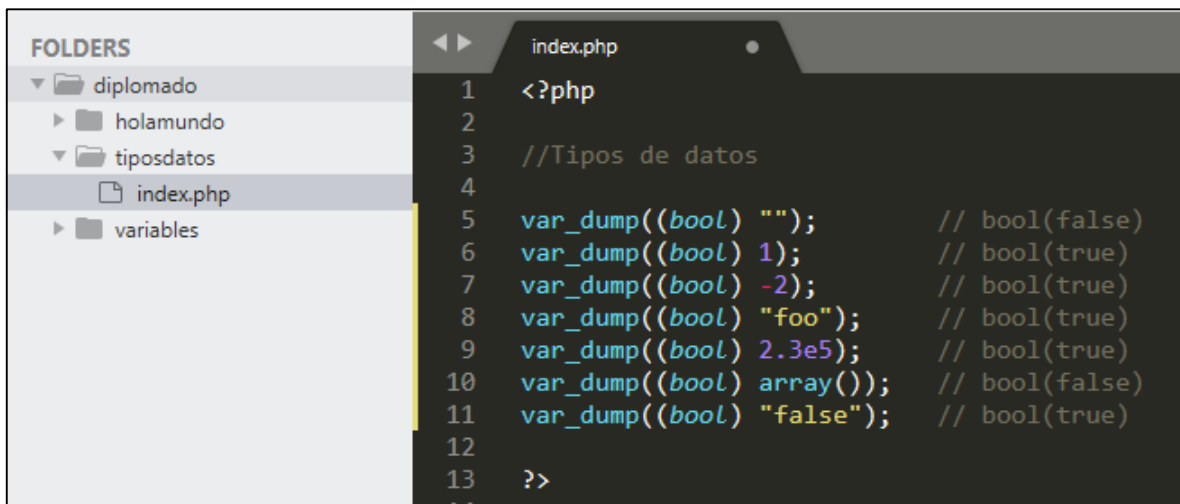
Para convertir explícitamente un valor al tipo *boolean* se usa (*bool*) o (*boolean*). Sin embargo, un valor será convertido automáticamente si un operador, función o estructura de control requiere un argumento de tipo *boolean*. Por ejemplo, se consideran falsos los siguientes valores:

- ✓ El *boolean FALSE* mismo.
- ✓ El *integer 0* (cero)

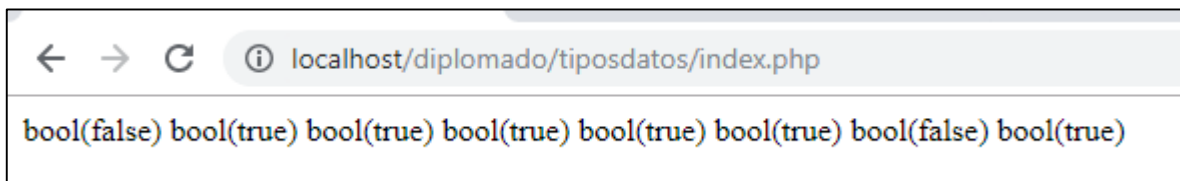
- ✓ El *float* 0.0 (cero)
- ✓ El valor *string* vacío, y el string "0"
- ✓ Un *array* con cero elementos
- ✓ Un *object* con cero variables miembro (solo en PHP 4)
- ✓ El tipo especial *NULL*.

Nota: -1 se considera TRUE, como cualquier otro número distinto de cero (ya sea negativo o positivo).

Algunos ejemplos de ejecución de conversión a *boolean* son los siguientes, en aplicación con el *var_dump*:



```
1 <?php
2
3 //Tipos de datos
4
5 var_dump((bool) "");           // bool(false)
6 var_dump((bool) 1);            // bool(true)
7 var_dump((bool) -2);           // bool(true)
8 var_dump((bool) "foo");        // bool(true)
9 var_dump((bool) 2.3e5);         // bool(true)
10 var_dump((bool) array());       // bool(false)
11 var_dump((bool) "false");       // bool(true)
12
13 ?>
```



localhost/diplomado/tiposdatos/index.php

bool(false) bool(true) bool(true) bool(true) bool(true) bool(true) bool(false) bool(true)


Ilustraciones 119 y 120.
Fuente: autor.

Arrays (arreglos)

El tipo de dato *array* representa una matriz donde pueden almacenarse múltiples valores en una sola variable asignada en una posición de la matriz.



```
1 <?php
2
3 //Tipos de datos
4
5 $ciudades = array("Medellín", "Bogota", "Cali", "Cartagena");
6
7 var_dump($ciudades);
8
9 ?>
```



```
array(4) { [0]=> string(9) "Medellín" [1]=> string(6) "Bogota" [2]=> string(4) "Cali" [3]=> string(9) "Cartagena" }
```

Ilustraciones 121 y 122.

Fuente: autor.

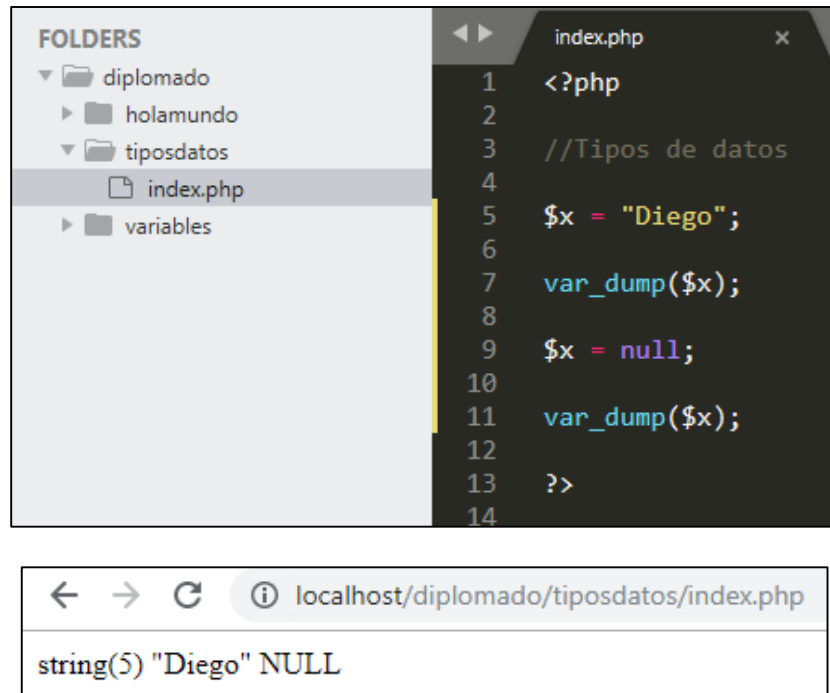
En el ejemplo anterior, en la declaración de *array* «Ciudades», esta toma todos los valores separados por coma como valores totalmente independientes, es decir, existen cuatro «variables» —valores en el *array*—. `var_dump` ayuda a ilustrar un poco la forma de almacenar valores en los *arrays*.

El concepto específico de los *arrays* será tratado a mayor profundidad en siguientes módulos, este es un pequeño abre bocas al funcionamiento de estos.

Null

Null es un tipo de datos especial que puede tener un solo valor: *null*. Una variable de tipo de datos *null* es una variable que no tiene ningún valor asignado.

Nota: cuando se crea una variable sin valor, se le asigna automáticamente *null*.



Ilustraciones 123 y 124.

Fuente: autor.



Ejercicio práctico

Ahora que realizaste tu primer proyecto en PHP sobre el Hola Mundo, conociste las primeras características del lenguaje y viste cómo se realiza el ejercicio en otros lenguajes, te pregunto:

¿Deseas profundizar en la temática de variables?

Entonces te sugiero realizar los siguientes ejercicios que pondrán a prueba los conocimientos adquiridos (en el módulo 1 - ejercicios de variables).

Existen otros tipos de datos que se deben tener en cuenta en el desarrollo bajo PHP, pero, para efectos del diplomado, serán abordados en el transcurso del diplomado, dado que algunos son un poco complejos y pueden generar confusiones en este momento de aprendizaje. Por otra parte, te invito a consultar un poco sobre: **gettype** y **settype**.

Realiza ejercicios con ambos:

Settype: <https://www.php.net/manual/es/function.settype.php>

Gettype: <https://www.php.net/manual/es/function.gettype.php>



Referencias bibliográficas

Apache Friends. (s.f.). *¿Qué es XAMPP?*

<https://www.apachefriends.org/es/index.html>

Lenguaje de programación. (2019, 12 de mayo). En *Wikipedia*.

https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n

PHP. (2022, 25 de abril). En *Wikipedia*.

https://es.wikipedia.org/wiki/PHP#Caracter%C3%ADsticas_de_PHP

Sublime Text. (2019, 16 de abril). En *Wikipedia*.

https://es.wikipedia.org/wiki/Sublime_Text

The PHP Group. (s.f.). *¿Qué es PHP?* My PHP.net.


<https://www.php.net/manual/es/intro-what-is.php>

The PHP Group. (2019). *Manual de PHP*. My PHP.net.

<https://www.php.net/manual/es/>

W3Schools. (2019, 28 de mayo). *PHP operators*.

https://www.w3schools.com/php7/php7_operators.asp



Esta guía fue elaborada para ser utilizada con fines didácticos como material de consulta de los participantes en el diplomado virtual en PROGRAMACIÓN EN PHP del Politécnico de Colombia, y solo podrá ser reproducida con esos fines. Por lo tanto, se agradece a los usuarios referirla en los escritos donde se utilice la información que aquí se presenta.

GUÍA DIDÁCTICA 1

M2-DV59-GU01

MÓDULO 1: CONCEPTOS BÁSICOS

© DERECHOS RESERVADOS - POLITÉCNICO DE COLOMBIA, 2023
Medellín, Colombia

Proceso: Gestión Académica Virtual
Realización del texto: Diego Palacio, docente
Revisión del texto: Comité de Revisión
Diseño: Comunicaciones

Editado por el Politécnico de Colombia.