

## 1 INTRODUCCION

---

Después de utilizar en mis cursos manuales de otros autores para dar clase de programación en php durante muchos años me voy a aventurar a realizar mi propio manual con los conceptos que utilizo normalmente en mis cursos.



Estos conceptos son los que a mí me resultan imprescindibles que un programador que quiera trabajar en php conozca como mínimo.

Voy a intentar que el manual contenga la teoría básica (al igual que realizo en mis clases) e indicaros donde podríamos buscar más información en caso de que necesitemos más información.

Además, el manual se complementará con ejercicios y practicas resueltas que nos permitirán ahondar en los términos explicados.

## 2 POR DONDE EMPEZAR

---

Lo primero que tenemos que tener claro es que es PHP.

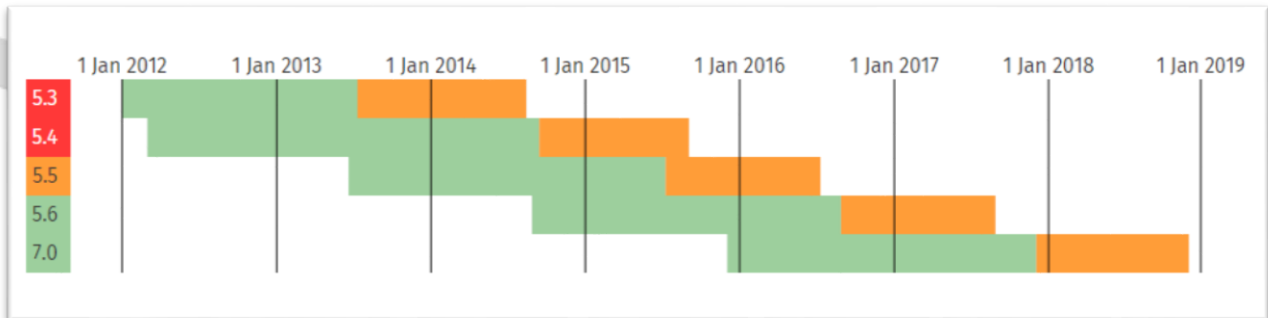
PHP es un lenguaje de programación a nivel de servidor.

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje de 'scripting' de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML.

Veamos un poco de historia que nunca está de más.

PHP es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

La siguiente gran contribución al lenguaje se realizó a mediados del 97 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP *versión 3*.



Posteriormente, con la *versión 4* de PHP se incluyó el motor Zend, que daba una mayor cobertura a las necesidades de ese momento y solucionaba problemas de la versión predecesora. Además se ganó en velocidad (compilando primero para después ejecutarse), así como en independencia del servidor web (con versiones nativas adaptadas a más plataformas) y con un API con más funciones y mucho más completo.

Fue con la *versión 5* cuando PHP alcanzó su punto álgido, al incorporar una mejor integración de la Programación Orientada a Objetos, que, aunque ya estaba disponible en la versión 4, no era capaz de cubrir las necesidades de los desarrolladores. Tal fue su importancia que estuvo durante más de once años en el mercado, estando actualmente en continuo mantenimiento.

Esa etapa dorada del PHP produjo grandes frutos, como la incorporación de numerosas herramientas que consiguieron responder a las demandas de los usuarios, como el autoloading de clases, que permitió incorporar el gestor de paquetes Composer. Todas estas mejoras permitieron a PHP equipararse a las herramientas ofertadas por otros lenguajes.

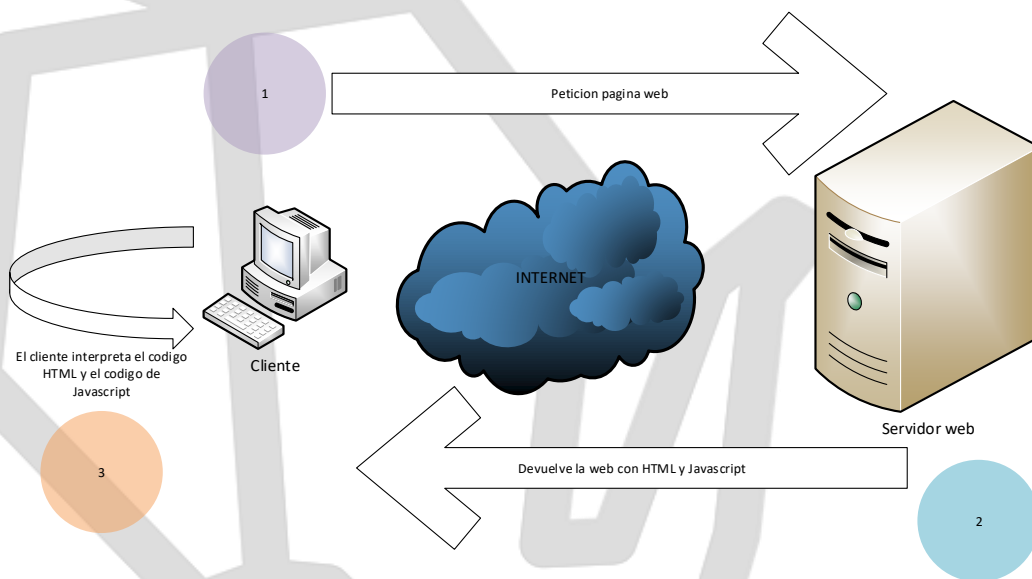
Tras 11 largos años llenos de cambios, de la versión PHP 5 se pasó directamente a la PHP 7, ya que la versión 6 no llegó a ver la luz por diversos inconvenientes.



Pues bien, con la *versión PHP 7* se incorporaron importantes mejoras en cuanto al rendimiento, que volvieron a situar a PHP entre los lenguajes más fuertes. Hoy en día está disponible en muchos servidores, pero no en todos, ya que al contar con tantas librerías y un software que no está completamente actualizado, arroja todavía errores al ejecutarse con esta versión.

## 2.1 Lenguajes de cliente y lenguajes de servidor

Vemos cómo funciona por ejemplo un lenguaje a nivel de cliente como Javascript o HTML.



Cuando tu solicitas la página al servidor, este te la devuelve y tu navegador es el encargado de analizar el código que viene en ella. Este código debe ser ejecutado por el navegador y por lo tanto este lo tiene que entender. A estos lenguajes se les denomina del lado del cliente. Podemos poner como ejemplo:

- HTML
- Javascript

El funcionamiento de PHP es distinto.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

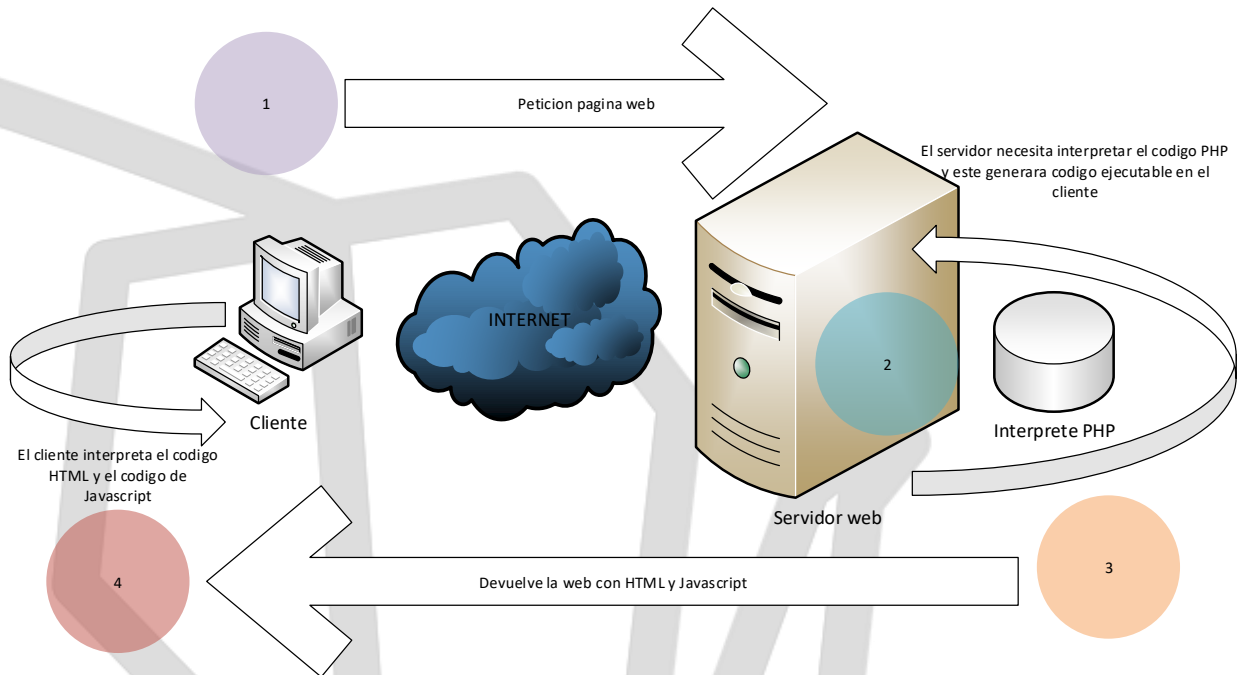
Para poder programar en PHP sería conveniente tener un servidor web instalado, además debemos tener el php instalado en el servidor web y para los temas de conexión con bases de datos necesitaremos un servidor de base de datos. En nuestro caso utilizaremos como servidor de base de datos MYSQL.

Luego como conclusión para poder empezar a trabajar necesitaremos:

- Servidor web: Vamos a utilizar Apache
- Interprete de Php instalado y configurado en el servidor.

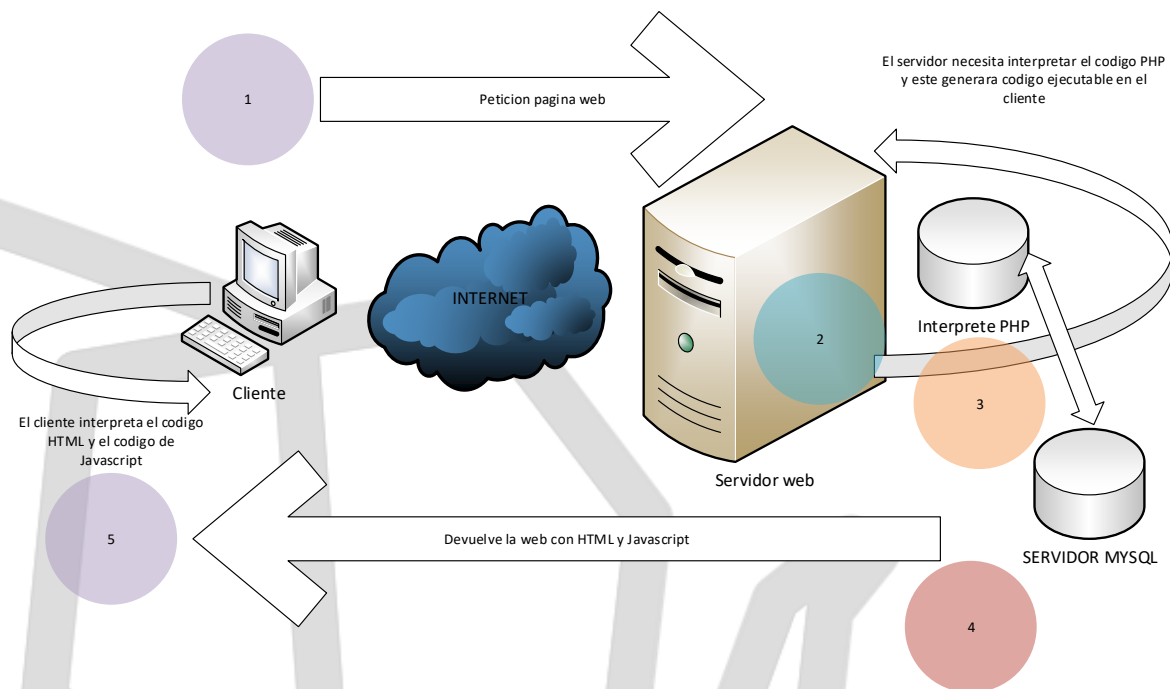
- Servidor de base de datos: En nuestro caso vamos a utilizar MYSQL.

Veamos el funcionamiento de una página PHP básica:



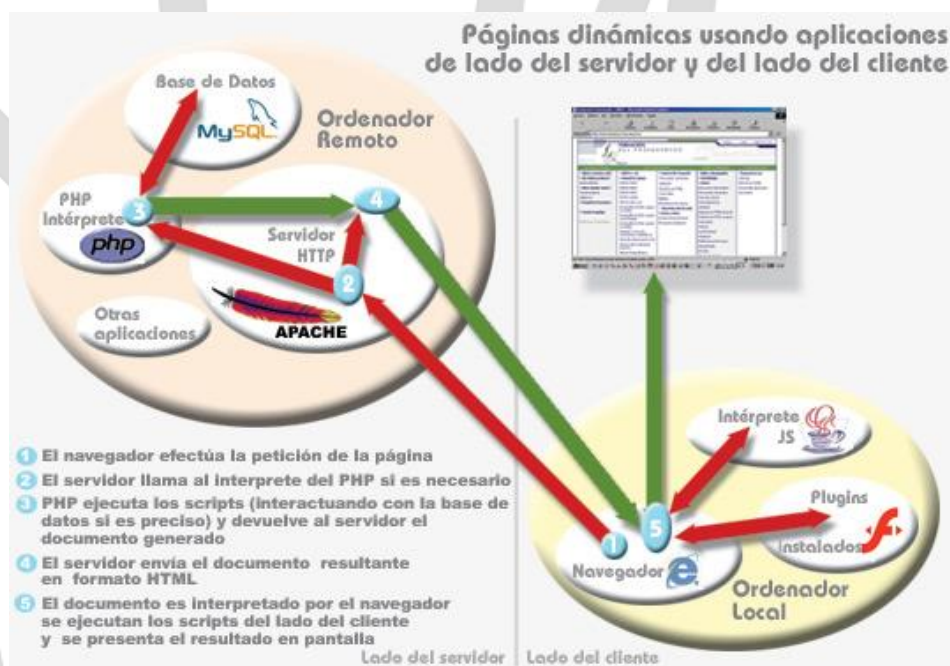
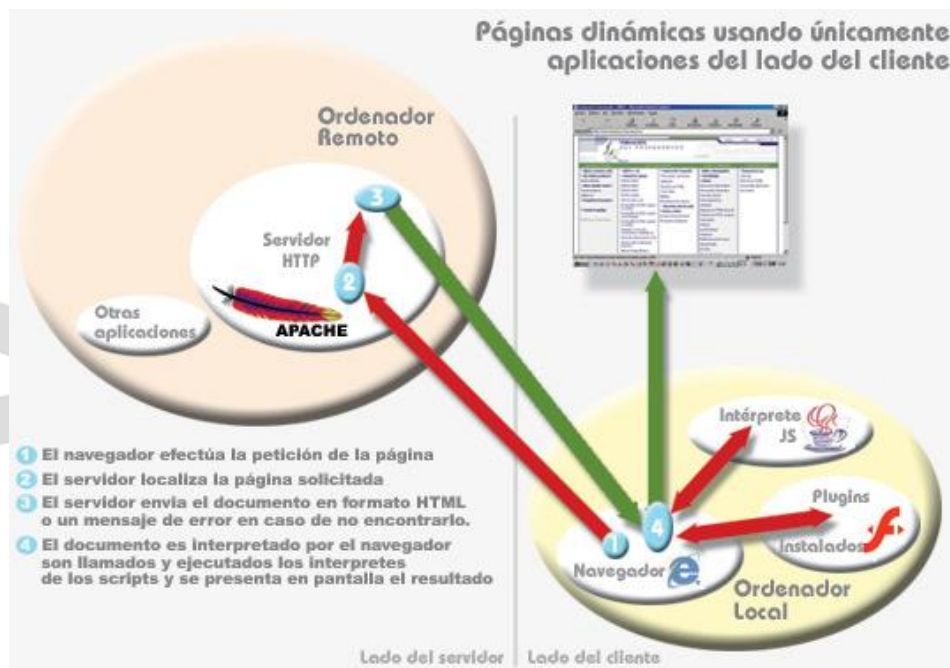
- El cliente solicita la página php al servidor web (1).
- El servidor debe analizar la página en busca de código PHP y cuando lo encuentre llamar al intérprete de PHP para que lo ejecute(2).
- Ahora el servidor devuelve la página web al cliente conteniendo únicamente contenido que este pueda ejecutar: HTML, Javascript, CSS (3).
- El cliente recoge esa información la interpreta y genera la web (4).

La página PHP puede necesitar datos de una base de datos y ahí es donde interviene nuestro servidor de base de datos. Incluyamos este elemento en nuestro esquema básico:



- El cliente solicita la página php al servidor web (1).
- El servidor debe analizar la página en busca de código PHP y cuando lo encuentre llamar al intérprete de PHP para que lo ejecute (2).
- El código PHP se encuentra con la necesidad de recolectar datos del servidor de bases de datos y para ello utilizara una extensión del lenguaje (3).
- Ahora el servidor devuelve la página web al cliente conteniendo únicamente contenido que este pueda ejecutar: HTML, Javascript, CSS (4).
- El cliente recoge esa información la interpreta y genera la web (5).

Siempre me han gustado dos esquemas que encontré navegando por Internet que relatan el funcionamiento de los lenguajes de servidor y de cliente:



Lo mejor para poder entender estos conceptos sería poner un par de ejemplos, pero antes de realizarlo vamos a ver que entorno tengo que preparar para poder ejecutar estos ejemplos.

### 3 Como preparamos nuestro ordenador para php

Para poder programar en php durante el transcurso de este manual vamos a necesitar los siguientes elementos:

- Servidor apache
- Interprete de php y configurarlo en apache.



- Servidor mysql y configuración básica
- Además, después necesitaremos añadir algunas extensiones a php para poder realizar ciertas tareas adicionales que iremos viendo a lo largo del curso

Para poder tener todo esto tenemos varias alternativas:

- Instalar a mano cada una de estas herramientas y después ir configurarlas
- Utilizar un paquete que nos instale y configure todas estas herramientas
- Utilizar un hosting
- Utilizar un entorno de virtualización.

En un principio nosotros vamos a utilizar uno de los paquetes que existen en el mercado para este propósito.

Además, necesitaremos un IDE que nos ayude en nuestra labor de programar en php. Para ello tenemos multitud en el mercado:

- Dreamweaver
- Eclipse
- Aptana
- Netbeans
- Sublime

Nosotros nos decantamos por el Netbeans.

### 3.1 Paquetes de todo en uno

La primera alternativa que sería recomendable para la mayoría de las personas es la utilización de un instalador todo en uno, que nos permite disponer del conjunto de programas necesarios, ya listos para utilizar.

Tanto en Windows como en Mac existen varios programas instaladores que nos pueden reducir tarea de crear el entorno de desarrollo a un cómodo asistente. Para Linux, aunque existen también alternativas en este sentido, no serían tan recomendables porque en este sistema lo ideal es instalar el software mediante los repositorios de cada distribución.

Estos programas instalan y configuran Apache + PHP + MySQL y además otros programas adicionales como pueden ser PhpMyAdmin, servidores de FTP y cosas similares, dependiendo de cada caso.

En Windows que es el sistema operativo con el que vamos a trabajar durante el curso:

- Xampp, el instalador más utilizado, es el que nosotros vamos a utilizar.

- Wamp. En caso que experimentemos cualquier problema con Xampp, Wampserver es una buena alternativa.

En Mac:

- Mamp, la primera opción para los desarrolladores de Mac.
- Mamp Pro. El propio Mamp tiene una versión profesional, de pago, pero que merece la pena si nos dedicamos profesionalmente al desarrollo web.

## 3.2 Instalar el xampp

XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MYSQL, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar.

El proceso es muy sencillo, primero debemos descargarnos el software de instalación.



Solamente hay que seguir los pasos en el proceso de instalación.

Durante la instalación nos pedirá si queremos que las herramientas Apache y Mysql sean servicios y se arranquen automáticamente o de lo contrario debemos arrancarlos manualmente a través del panel de control de Xampp.

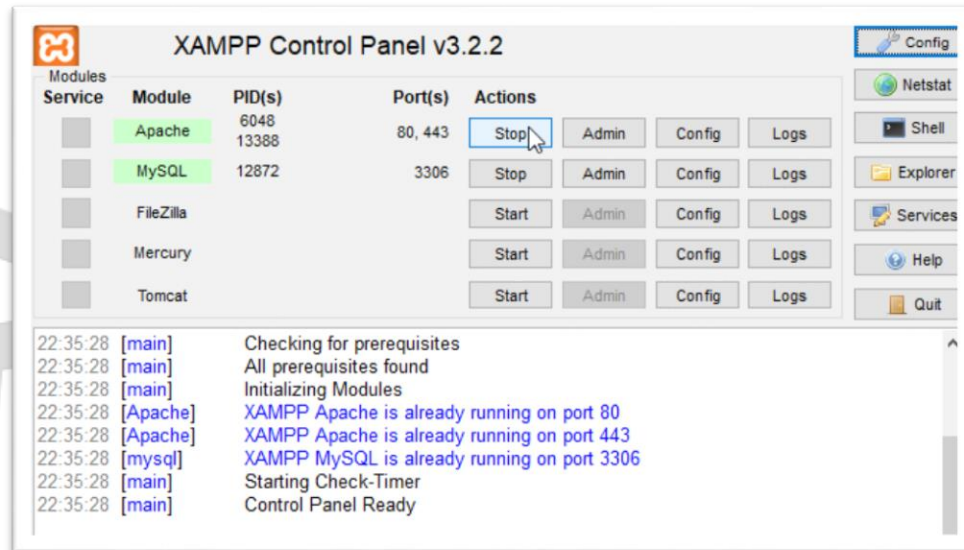
Nosotros hemos decidido no instalarlo como servicio y por lo tanto cada vez que queremos programar realizamos lo siguiente:

- 1- Arrancar el panel de control de Xampp





## 2- Arrancamos por lo menos Apache y MYSQL



Para comprobar que funciona correctamente simplemente debéis arrancar el navegador (nosotros vamos a utilizar como navegadores el Chrome y el Firefox).

Escribir directamente en la barra de direcciones:

- 127.0.0.1 o localhost

Debería salir la página de Xampp.

## 3.3 Utilizar un entorno de virtualización

Otra opción es crear un entorno de desarrollo virtual que sea fácilmente reproducible por todos los desarrolladores, de tal forma que no tengamos el típico problema de ¿En mi ordenador funcionaba bien?

Este concepto es un poco complicado de entender al principio pero voy a tratar de poner un ejemplo.

Supongamos que un equipo de trabajo de 5 personas, quiere desarrollar un sistema web (o una página web), la cual correrá en un servidor bajo un Apache con una base de datos MySQL. Este servidor, será un servidor externo al que se le hará una petición y el servidor dará una respuesta (por ejemplo el hosting de una empresa). Teneis que tener en cuenta que este hosting tendrá unas características determinadas que tendremos que fusilar en nuestro ambiente de desarrollo.

Suponeros que en este equipo de trabajo cada uno tiene en su PC un sistema operativo distinto (Un OS, un Debian, un Ubuntu, un Windows, ...) y en este caso cada uno tendría que tener en su PC instalado el Apache con su MySQL y demás, al igual que tendría el servidor al que se subiría el código de la web.

Como esto puede ser muy tedioso y además a veces complicado lo suyo sería tener clonado en el PC de cada uno de los integrantes del equipo de trabajo una máquina virtual con un entorno igual al que tendremos en nuestro servidor donde correrá nuestra página web.

Una opción sería crear una máquina virtual con cualquiera de los proveedores de máquinas virtuales que hay en el mercado (VirtualBox, VmWare,...) y después pasare a cada miembro del equipo esa máquina.

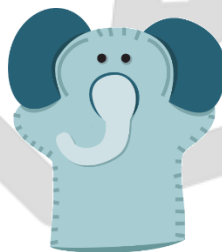
Esto además de ser un trabajo pesado puede dar también problemas que tendríamos que solucionar y nos llevaría un tiempo precioso.

Para ello podemos utilizar muchas herramientas. Yo principalmente os aconsejaría un par de ellas:

- Vagrant : nos permite crear una maquina virtual a través de un fichero de configuración en un proveedor de maquinas virtuales.



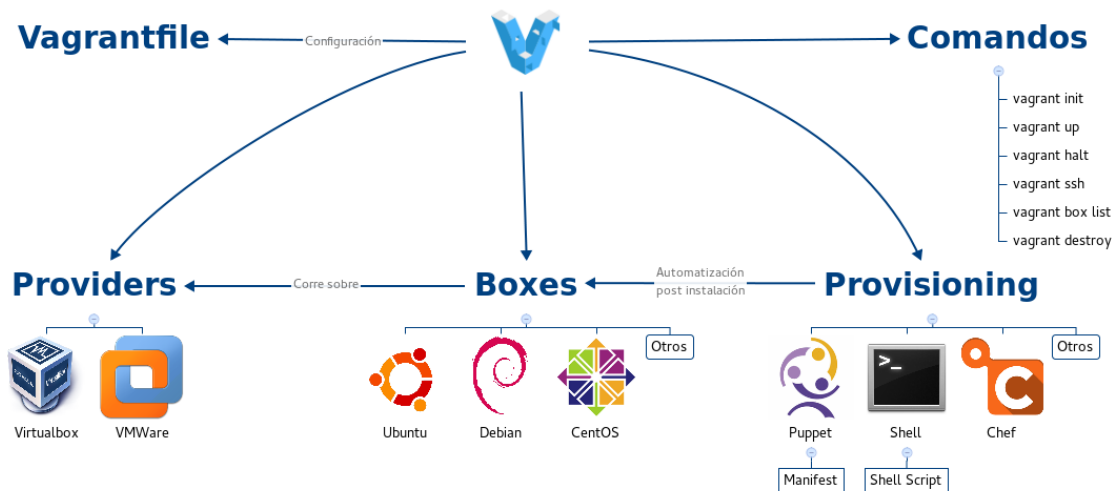
- PuPHPet : nos permite configurar el fichero de configuración para vagrant de tal forma que podemos seleccionar el entorno de desarrollo o pruebas a generar en la maquina virtual.



### 3.4 Vagrant

Vagrant es una herramienta que se va colocar entre nosotros y el software de maquina virtual que utilicemos para poder crearla y configurarla de forma declarativa (mediante comandos), de tal forma que nos va ser muy fácil la portabilidad de estas.





Hay algunos conceptos que debemos tener claros antes de seguir:

- Proveedor de máquina virtual : es el software para crear máquinas virtuales, en nuestros caso VirtualBox.
- Boxes: Son los paquetes de vagrant que vamos a descargar para generar las máquinas virtuales.
- Provisionamiento: Son herramientas que nos van a permitir mediante comandos añadir y configurar herramientas en nuestros equipos virtuales. Por ejemplo, instalar el Git, instalar y configurar apache, ...

El que haya tenido que realizar alguna vez maquina virtuales sabemos que es un proceso largo y repetitivo:

- Colocarle bien los requisitos de memoria
- Buscar y montar las ISO
- Montar las unidades.
- ...

Vagrant nos va a ayudar en estas tareas para poderlas realizar mediante comandos y por la tanto de una forma mas automática.

Vagrant es una herramienta gratuita de línea de comandos, disponible para Windows, MacOS X y GNU/Linux, que permite generar entornos de desarrollo reproducibles y compartibles de forma muy sencilla.

Para ello, Vagrant crea y configura máquinas virtuales a partir de simples ficheros de configuración.

Lo que nos va a permitir por lo tanto es a través de un archivo de comandos levantar una máquina virtual para crear un entorno de pruebas o de desarrollo.

Basta con compartir el fichero de configuración de Vagrant (llamado "Vagrantfile") con otro desarrollador para que, con un simple comando, pueda reproducir el mismo entorno de desarrollo. Esto es especialmente útil en equipos formados por varias personas, ya que asegura que todos los desarrolladores

tienen el mismo entorno, con las mismas dependencias y configuración. Con Vagrant, compartir pesadas máquinas virtuales o el ya mítico “en mi ordenador funciona” (causado generalmente por diferentes configuraciones o versiones de software) son cosas del pasado.

Además, dado que la configuración de la máquina virtual es un simple fichero de texto plano, podemos incluir este fichero en nuestro repositorio en el control de versiones, junto con el resto del código del proyecto. De esta manera, un nuevo desarrollador que se incorpore al equipo simplemente tendrá que clonar el repositorio del proyecto y ejecutar Vagrant para tener el entorno de desarrollo montado y funcionando en cuestión de minutos.

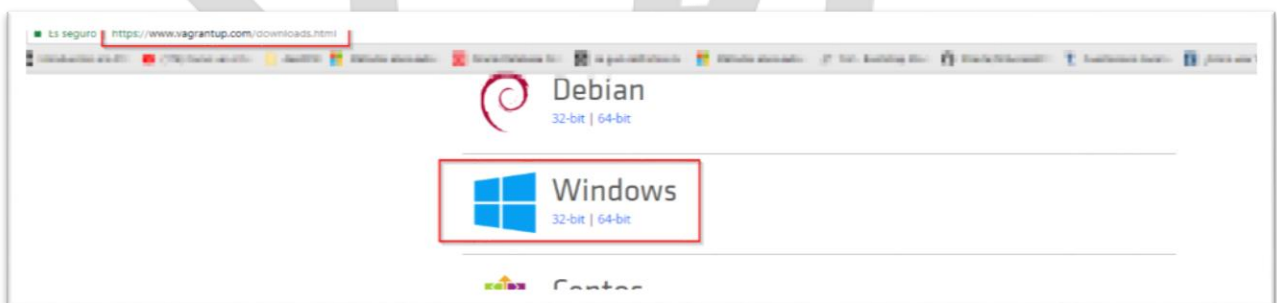
Por defecto, Vagrant utiliza VirtualBox como motor de máquinas virtuales, aunque existe la opción de utilizar VMWare Workstation (Windows) o VMWare Fusion (MacOS X) con un plugin de pago.

### 3.5 Primeros pasos con Vagrant

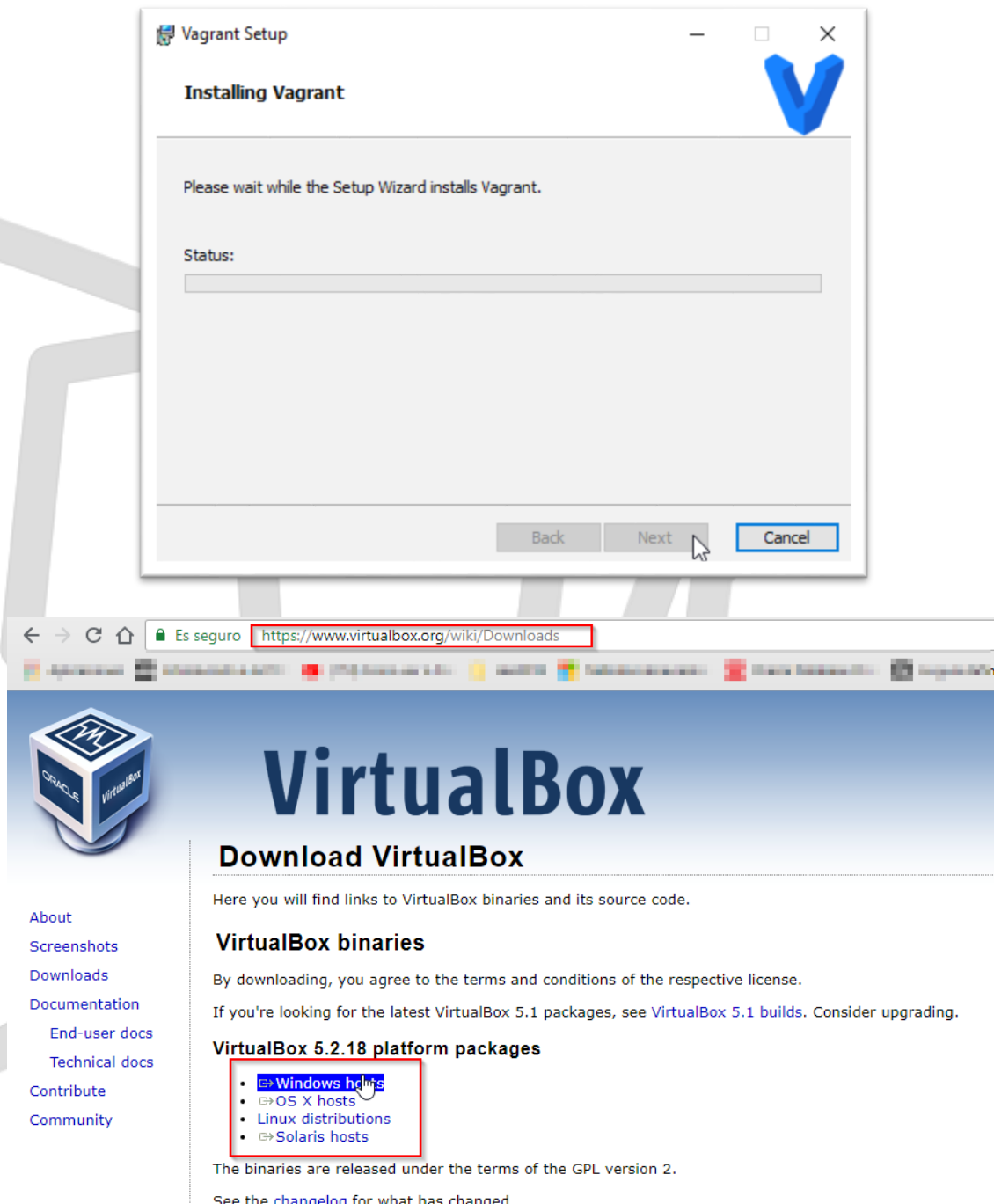
#### *Descargar e instalar Vagrant y VirtualBox*

El primer paso, como es habitual, es descargar e instalar Vagrant y además instalar el proveedor de máquinas virtuales que queramos utilizar, que por defecto será VirtualBox, ya que es gratuito y viene integrado en Vagrant.

Para descargar Vagrant nos vamos a su web y buscamos la versión para nuestro sistema operativo.



Ejecutamos el instalador y esperamos a que termine.



### Como funciona esto

El funcionamiento básico es el siguiente:

- Debemos descargarnos un box con toda la máquina virtual que yo quiera utilizar. Estos box están en internet.
- Ahora crearemos una carpeta donde vamos a crear un fichero vagrantfile para colocar las configuraciones de nuestra maquina

- Posteriormente debemos crear la máquina virtual mediante comandos de vagrant utilizando el fichero de configuración y el box descargado

Para mas información

<https://www.vagrantup.com/intro/getting-started/>

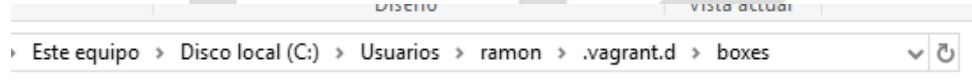
#### *Donde esta cada cosa:*

- Las máquinas virtuales se generarán en Windows en la carpeta



> Este equipo > Disco local (C:) > Usuarios > ramon > VirtualBox VMs >

- Los boxes de Vagrant se generarán en Windows en la carpeta



> Este equipo > Disco local (C:) > Usuarios > ramon > .vagrant.d > boxes

#### *Y ahora*

Una vez instalados, podremos ejecutar el comando 'vagrant' para obtener un listado de las opciones disponibles a través de una consola de comandos, podemos utilizar:

- Cmd
- Powershell
- Si teneis el git instalado, el propio git Shell (este es el que yo voy a utilizar, pero vale cualquiera).

Crear una máquina virtual con Vagrant es tan sencillo como ejecutar los siguientes comandos:

- Vagrant init: creas un archivo vagrantfile vacio
- Vagrant init ramon/maquin : crear un vagrantfile apuntando a un box concreto simplemente se trata de colocar la url al box al final.
- Vagrant up : arranca una maquina virtual , para ello lógicamente debe haber en el directorio un fichero vagrantfile
- Vagrant suspend: Guarda el estado de la máquina virtual y temporalmente cierra la máquina virtual. Si volvemos a ejecutar vagrant up la máquina virtual se restaurará rápidamente en el punto donde la dejamos.



- Vagrant ssh : Entrar en la máquina virtual que tienes arrancada usando ssh. Vagrant usa automáticamente sus claves ssh y las copia en la máquina virtual, es por eso que no es necesario autenticarse con usuario y contraseña
- Vagrant provision : Vuelve a ejecutar tu script de arranque (si lo has actualizado) sin tener que crear la máquina virtual desde cero.
- Vagrant reload: Resetea la máquina virtual y vuelve a lanzar el script de arranque
- Vagrant destroy: borra la máquina y el disco creado, aunque conserva el box
- Vagrant box add ramon/maqui : crear un box desde la url colocada al final (no genera el vagrantfile)
- Vagrant box list : ver todos los box instalados
- Vagrant box remove nombreBox: Elimina el box del ordenador.

Para ver todos los comandos yo os recomiendo ir a la url

<https://www.vagrantup.com/docs/cli/>

### *Añadir aplicaciones a la máquina virtual (provisionamiento)*

Para poder añadir aplicaciones a la máquina virtual podemos utilizar una Shell a la máquina virtual o una mejor decisión para que esto quedase almacenado utilizar el fichero vagrantfile.

En ese fichero indicaremos que queremos instalar y para ello utilizaremos herramientas como:

- Docker
- Chef
- Puppet
- Ansible
- Directamente a través de Shell scripts

Este concepto de añadir aplicaciones a vagrant es lo que se denomina APROVISIONAMIENTO

## 3.6 Donde puedo encontrar box

Para poder encontrar box podéis utilizar las siguientes direcciones:

*Oficial*

<https://app.vagrantup.com/boxes/search>

*Otra oficial*

<https://app.vagrantup.com/bento>

*Segura, pero no oficial*

<https://www.vagrantbox.es/>

### 3.7 Pasos detallados con Vagrant

#### *Instalar la maquina virtual*

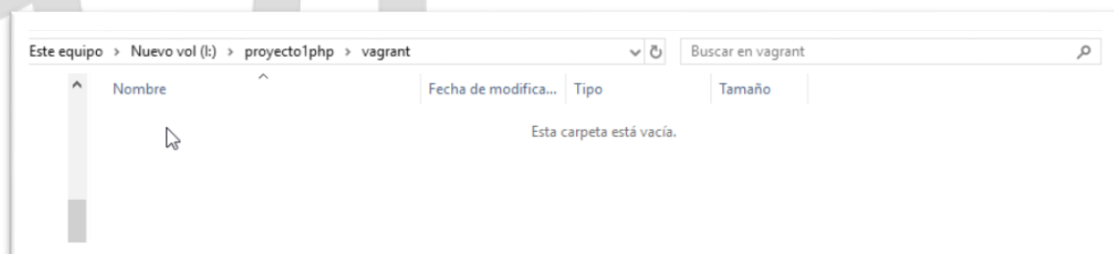
En este punto vamos a preparar un entorno para programar en php. Voy a instalar un entorno bajo Linux con Apache, Mysql y PHP.

Ademas vamos a colocar una carpeta compartida en nuestro ordenador para que este simulando la carpeta www de apache.

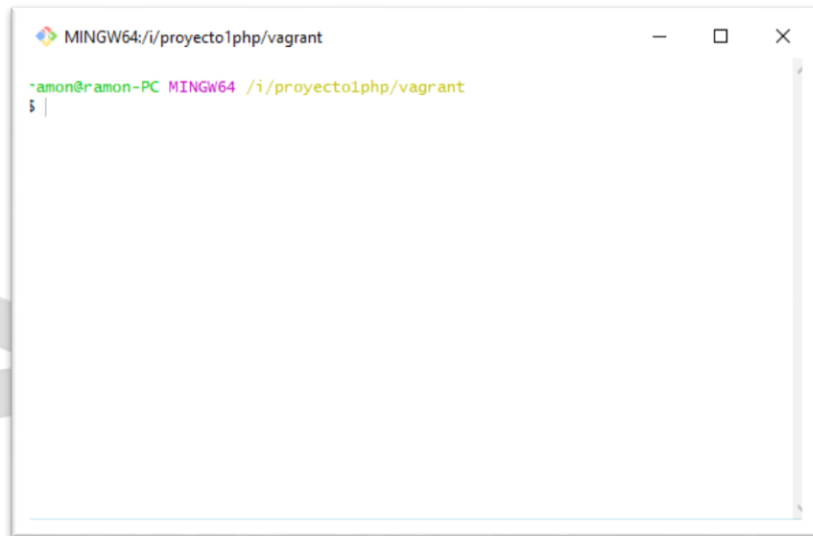
La primera pregunta es ¿Dónde me coloco para empezar?

Yo normalmente la maquina virtual la creo en mi propio proyecto web para que asi al subirlo al repositorio me llevo también la configuración de la maquina con el box y el aprovisionamiento.

Yo para mi ejemplo he creado una carpeta denominada proyecto1php en la unidad I de mi ordenador. Dentro de ella creo una carpeta denominada vagrant



Ahora abro en esta carpeta un terminal de git.



```
MINGW64:/i/proyecto1php/vagrant  
- - -  
*amon@ramon-PC MINGW64 /i/proyecto1php/vagrant  
$ |
```

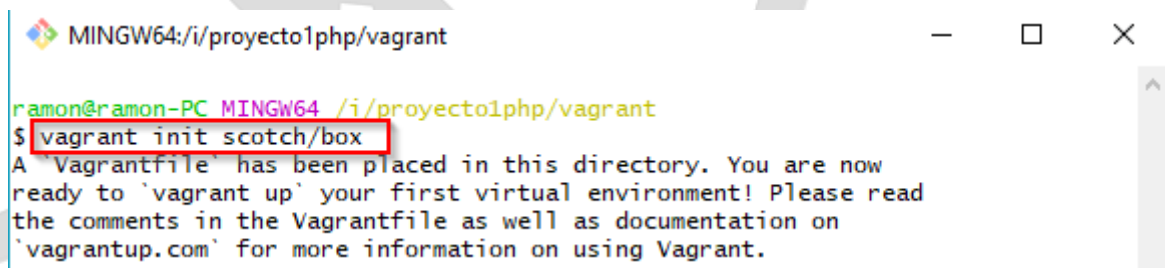
Si no tenéis git, podéis utilizar cmd o powershell. También podéis instalar git.

Ahora necesitamos crear el fichero vagrantfile. Para realizarlo podemos:

- Realizar el fichero vagrantfile con un editor de texto
- Vagrant init url realiza el fichero automaticamente

Vamos a utilizar scotch/box.

- Creo el fichero vagrantfile



```
MINGW64:/i/proyecto1php/vagrant  
- - -  
ramon@ramon-PC MINGW64 /i/proyecto1php/vagrant  
$ vagrant init scotch/box  
A 'Vagrantfile' has been placed in this directory. You are now  
ready to 'vagrant up' your first virtual environment! Please read  
the comments in the Vagrantfile as well as documentation on  
'vagrantup.com' for more information on using Vagrant.
```

- Veamos el fichero

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in
# Vagrant.configure
# configures the configuration version (we support older styles
# for
# backwards compatibility). Please don't change it unless you
# know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and
  # commented below.
  # For a complete reference, please see the online
  # documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You
  # can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "scotch/box"
```

Realmente con esto nos hubiera valido

```
Vagrantfile New
Vagrant.configure("2") do |config|
  config.vm.box = "scotch/box"
end
```

El resto son opciones de configuración y aprovisionamiento.

#### 4 ¿Qué nos permite realizar php?

Poco a poco el PHP se va convirtiendo en un lenguaje que nos permite hacer de todo. En un principio diseñado para realizar poco más que un contador y un libro de visitas, PHP ha experimentado en poco tiempo una verdadera revolución y, a partir de sus funciones, en estos momentos se pueden realizar una multitud de tareas útiles para el desarrollo del web:

##### *Funciones de correo electrónico*

Podemos con una facilidad asombrosa enviar un e-mail a una persona o lista parametrizando toda una serie de aspectos tales como el e-mail de procedencia, asunto, persona a responder...

##### *Gestión de bases de datos*

Resulta difícil concebir un sitio actual, potente y rico en contenido que no es gestionado por una base de datos. El lenguaje PHP ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft, a partir de las cuales podremos editar el contenido de nuestro sitio con absoluta sencillez.

##### *Gestión de archivos*

Crear, borrar, mover, modificar...cualquier tipo de operación más o menos razonable que se nos pueda ocurrir puede ser realizada a partir de una amplia librería de funciones para la gestión de archivos por PHP. También podemos transferir archivos por FTP a partir de sentencias en nuestro código, protocolo para el cual PHP ha previsto también gran cantidad de funciones. Tener en cuenta que estamos hablando de archivos del servidor

### Tratamiento de imágenes

Evidentemente resulta mucho más sencillo utilizar Photoshop para el tratamiento de imágenes, pero... ¿Y si tenemos que tratar miles de imágenes enviadas por nuestros internautas? La verdad es que puede resultar muy tedioso uniformar en tamaño y formato miles de imágenes recibidas día tras día. Todo esto puede ser también automatizado eficazmente mediante PHP.

### Generación de documentos PDF

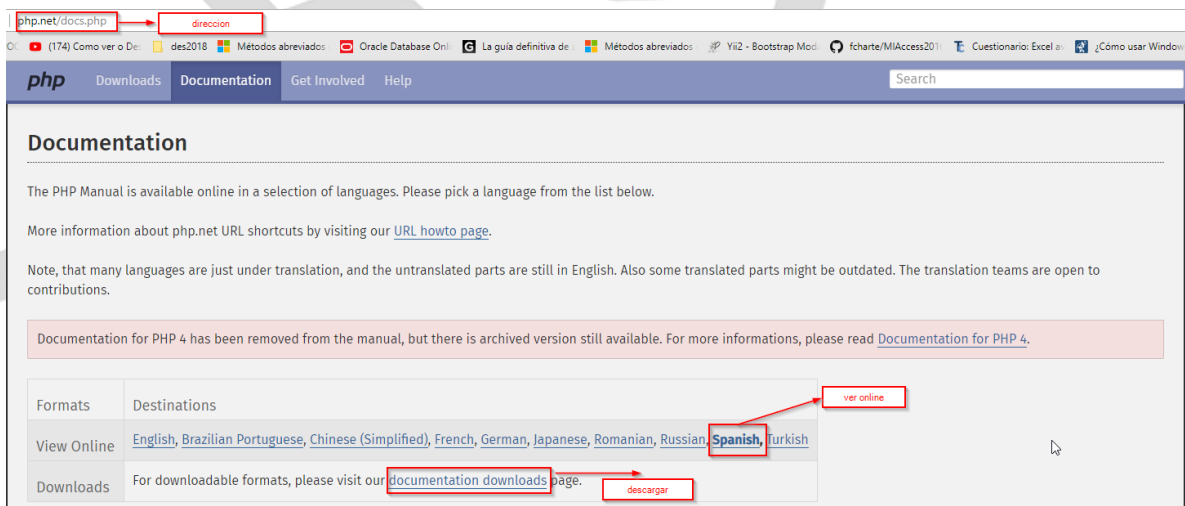
Podemos crear documentos PDF utilizando el contenido de la página a través de múltiples extensiones existentes en el mercado.

## 5 ¿Y si quiero algo más?

Como ya os he comentado en el manual yo voy a ir comentando la teoría básica para poder entender los principales conceptos de PHP. Además os indicare de donde podéis recolectar mas información.

Hay dos fuentes que os voy a adelantar como genéricas:

- La web de php



- La web de w3schools

← → ↻ 🔍 [https://www.w3schools.com/php/php\\_ref\\_overview.asp](https://www.w3schools.com/php/php_ref_overview.asp) dirección

Aplicaciones Introducción a la POI (174) Como ver o De des2018 Métodos abreviados Oracle Database Onli La guía definitiva de Métodos abreviados Yi2 - Bootstrap Mod fcharte/MIAccess201 Cuestionario: Excel a

HTML CSS JAVASCRIPT SQL **PHP** BOOTSTRAP HOW TO JQUERY W3.CSS PYTHON XML MORE REFERENCES

MySQL Update Data  
MySQL Limit Data

**PHP - XML**  
PHP XML Parsers  
PHP SimpleXML Parser  
PHP SimpleXML - Get  
PHP XML Expat  
PHP XML DOM

**PHP - AJAX**  
AJAX Intro  
AJAX PHP  
AJAX Database  
AJAX XML  
AJAX Live Search  
AJAX RSS Reader  
AJAX Poll

**PHP Examples**  
PHP Examples  
PHP Quiz  
PHP Certificate

**PHP Reference**  
PHP Overview  
PHP Array  
PHP Calendar  
PHP Date  
PHP Directory

This section contains a complete **PHP reference documentation.**

## PHP Reference

The PHP reference contains different categories of all PHP functions and constants, along with examples.

enlaces

Array	Calendar	Date	Directory	Error
Filesystem	Filter	FTP	HTTP	libxml
Mail	Math	Misc	MySQLi	SimpleXML
String	XML	Zip	Timezones	

Previous Next