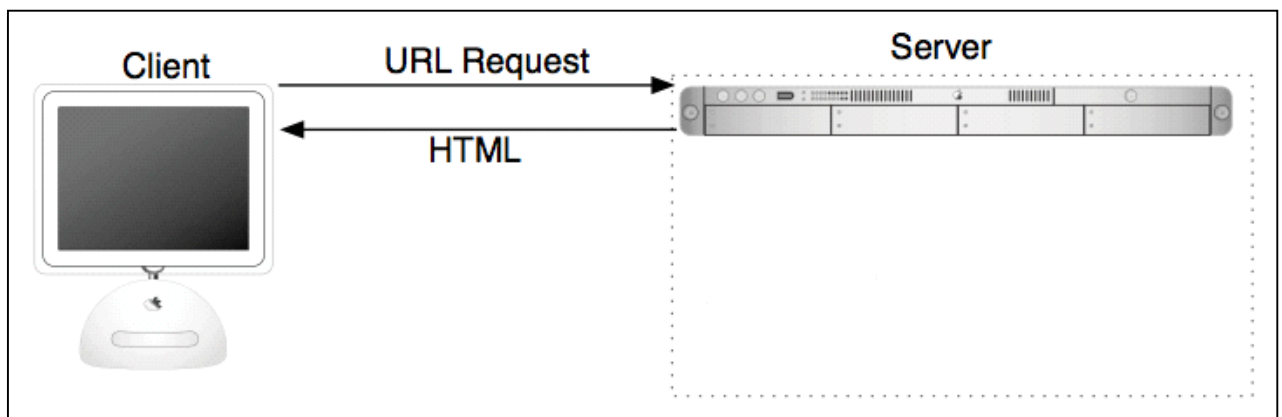


PHP. Un lenguaje de scripts de servidor.

01. Que es un lenguaje de script de servidor y cómo se usa

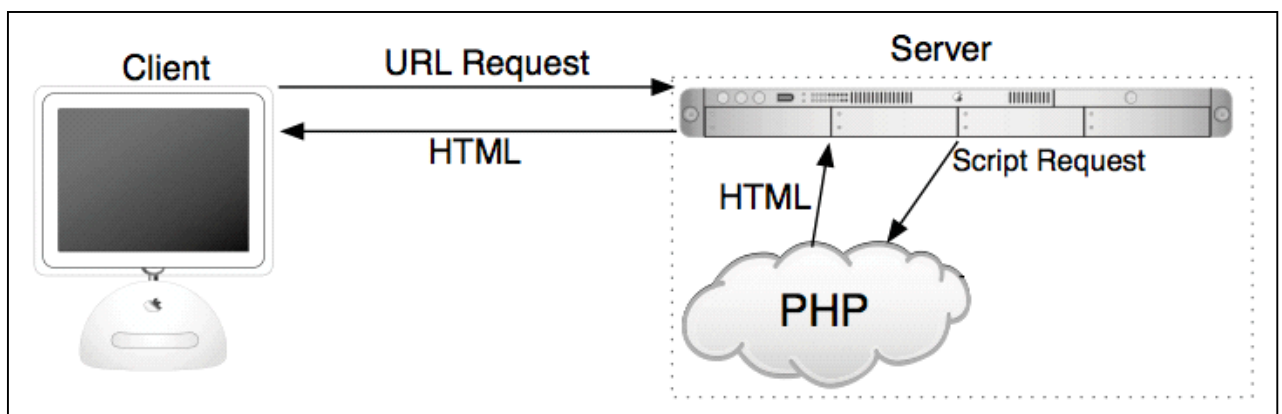
PHP es un lenguaje de script de servidor (no de cliente), es decir que es interpretado por la máquina que ofrece los servicios de páginas web. Éste se coloca dentro del código de una página web, es decir que el código de PHP estará dentro del código HTML de una página.

Ustedes ya han trabajado con HTML, el lenguaje utilizado para la creación de páginas web estáticas, la secuencia que se realiza para el procesamiento de una página web estática es similar al que se refleja en la siguiente figura:



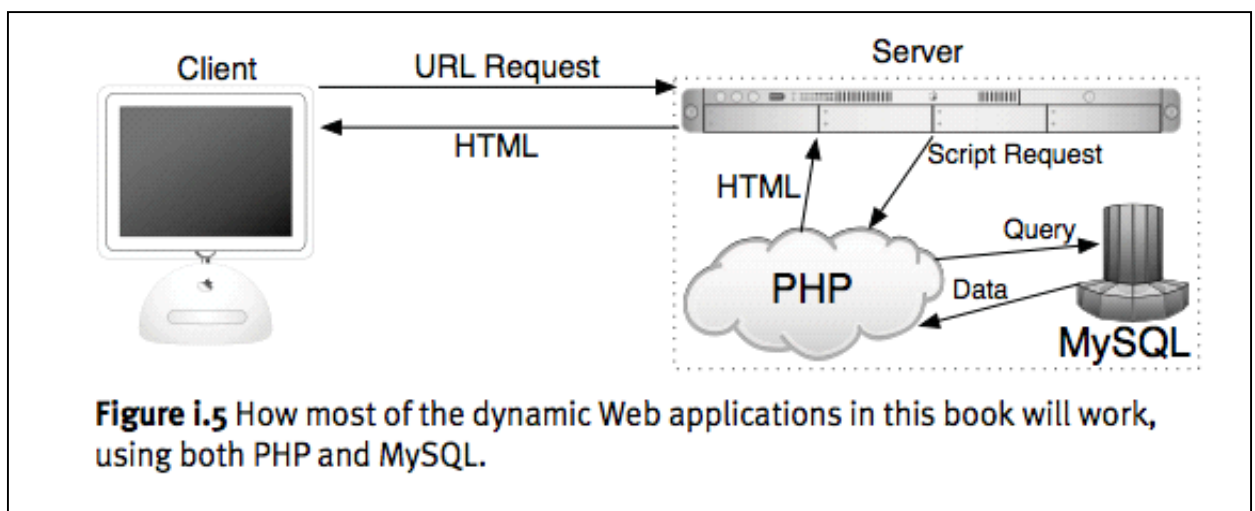
En ella se aprecia que el cliente a través de su navegador hace una petición de una página web al servidor, si esta página sólo contiene código HTML es enviada directamente al usuario y su navegador (programa encargado de procesar el código HTML) lo interpreta mostrando la página web solicitada.

Veamos ahora como será el procesamiento de una página web dinámica (que tiene código script que debe ser procesado por el servidor),



Ahora nuevamente el usuario hace una petición de una página web al servidor, esta página que ahora tiene código PHP será procesada por el servidor para ejecutar todo el código PHP que en ella se encuentre. Los resultados que se obtengan de la ejecución del script por parte del servidor es escrito en HTML. Una vez que el servidor haya procesado todos los scripts envía el código HTML resultante al navegador del usuario que lo procesa mostrando la página web solicitada.

Si la página web que se solicita hace uso de una base de datos (login de usuario, tienda virtual, etc.) entonces el diagrama anterior se modificaría quedando de la siguiente forma:



Ahora cuando el servidor procesa el script y se da cuenta que tiene que hacer uso de la base de datos, hace una petición al servidor de base de datos y este le devuelve los datos pedidos, que ahora serán integrados en el script y transformados en código HTML que será lo que se envía al navegador del usuario para su procesamiento.

02. Preparación de nuestro espacio de trabajo.

En este punto instalaremos y configuraremos los programas que nos permiten montar un servidor web de trabajo. Los programas que debemos tener son:

- Un servidor web (en nuestro caso será Apache)
- Un servidor de base de datos (en nuestro caso será MySQL)
- Un lenguaje de script de servidor (en nuestro caso será PHP)

Como lo que queremos montar es un servidor de trabajo, existen en el mercado una serie de programas que nos permiten hacer uso de Apache, MySQL y PHP sin tener

que instalarlos. Uno de los mejores es XAMPP y es el que usaremos para nuestro trabajo con PHP.

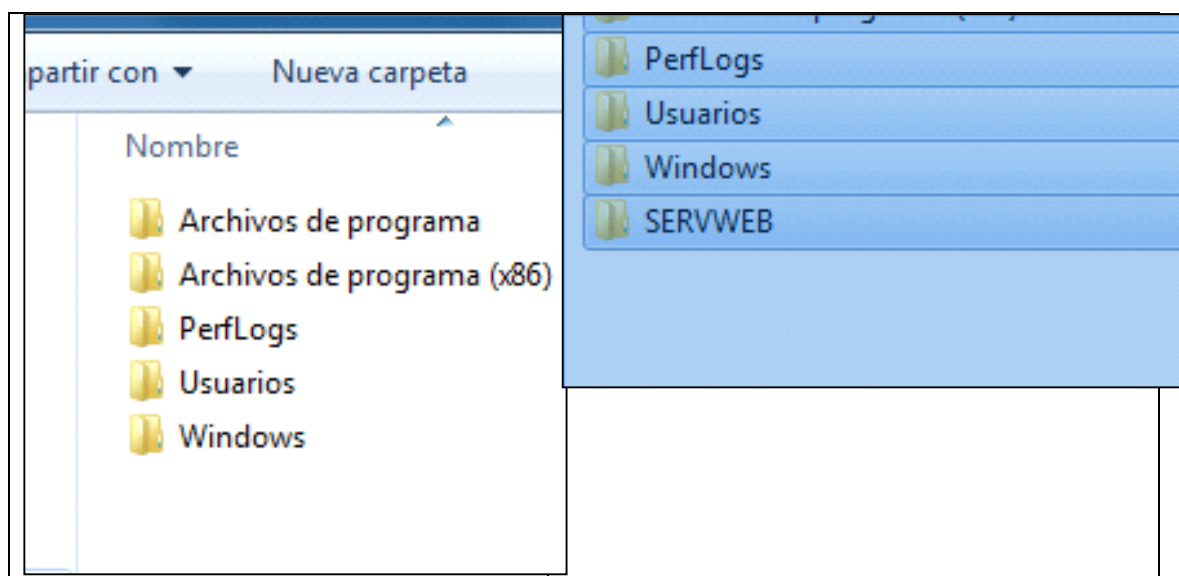
La versión de XAMPP que usaremos es la que no necesita instalación. La versión que usaremos es: **xampp-win32-1.8.0-VC9**.

Para su instalación debemos realizar los siguientes pasos:

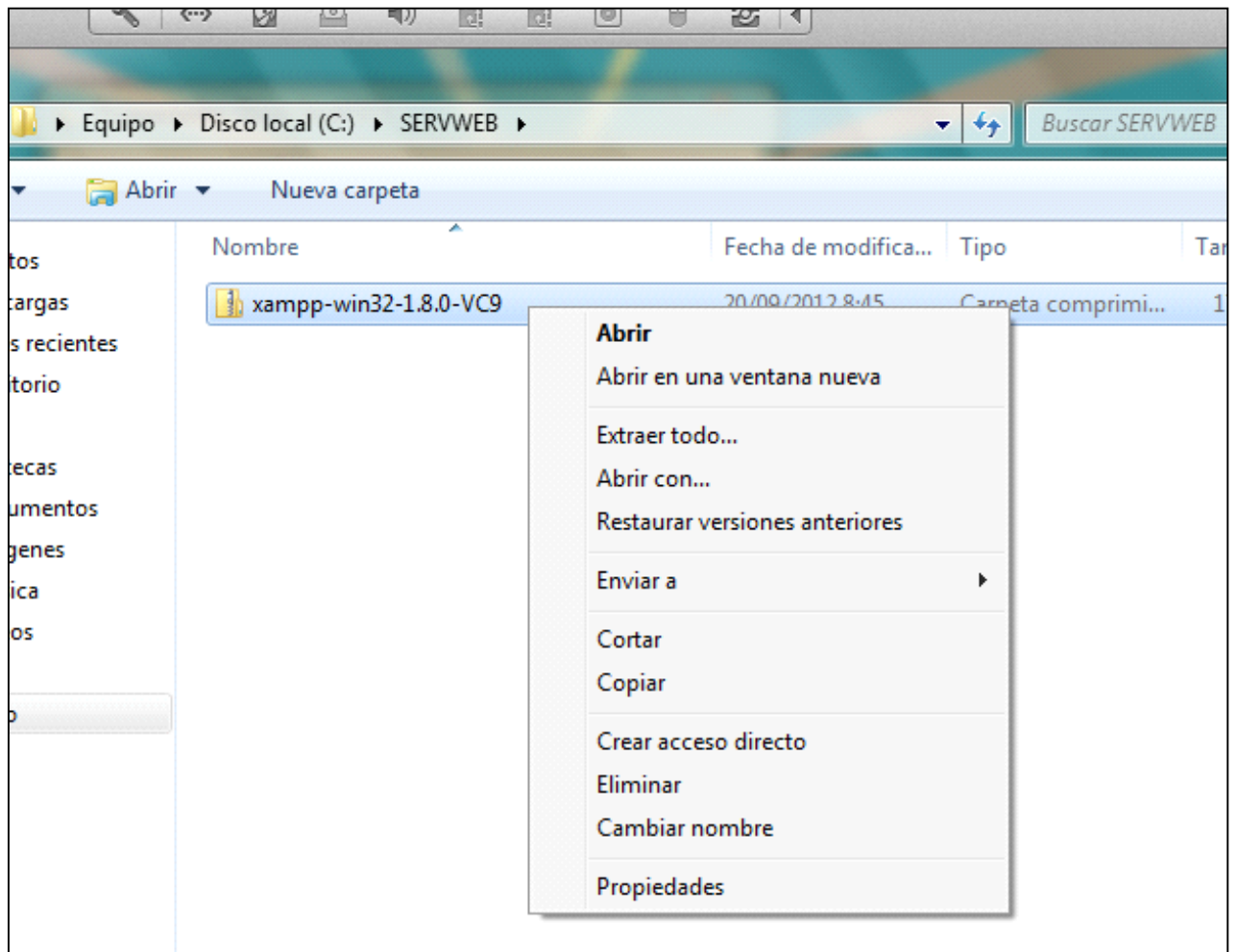
- Descargar XAMPP de su página oficial
<http://www.apachefriends.org/download.php?xampp-win32-1.8.0-VC9.zip>
- Crear una carpeta en la unidad C:\ de nombre SERVWEB.
- Descomprimir el contenido del fichero descargado en el punto 1 en la carpeta creada en el punto 2.
- Ejecutar el fichero "setup_xampp.bat" para ajustar la configuración de la aplicación con la ruta que hemos utilizado para descomprimirla.
- Ejecutar el fichero \xampp\xampp-control.exe para iniciar el panel de control de XAMPP.
- Iniciar el servicio de Apache.
- Verificar en un navegador que php funcione correctamente.
- Habilitar el servicio de MySQL.
- Verificar que phpmyAdmin funcione correctamente.
- Ejecutar configuraciones adicionales de seguridad.

Realicemos la instalación paso por paso después de haber descargado el fichero de XAMPP.

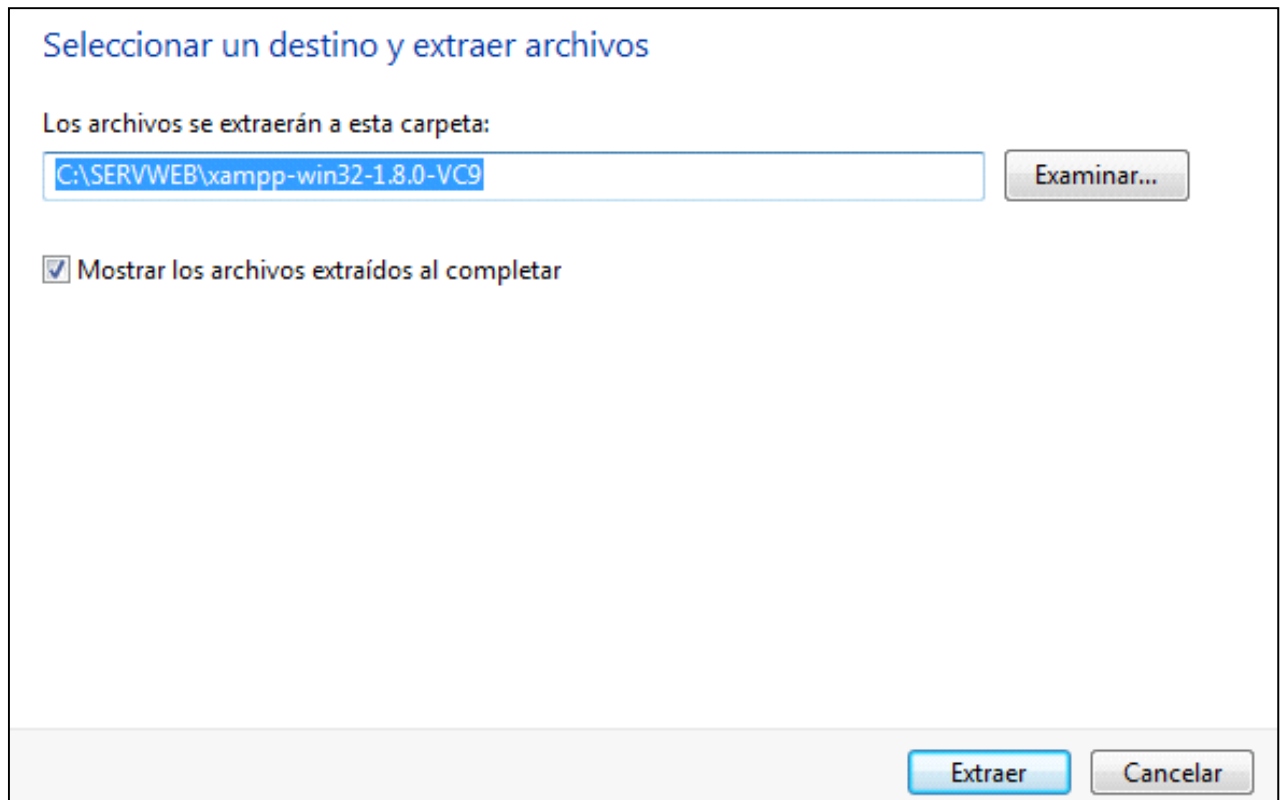
Creamos la carpeta SERVWEB



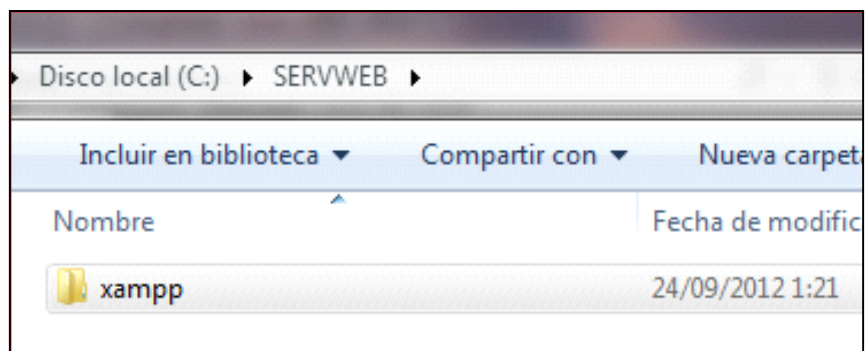
Descomprimos el fichero de XAMPP dentro de la carpeta SERVWEB



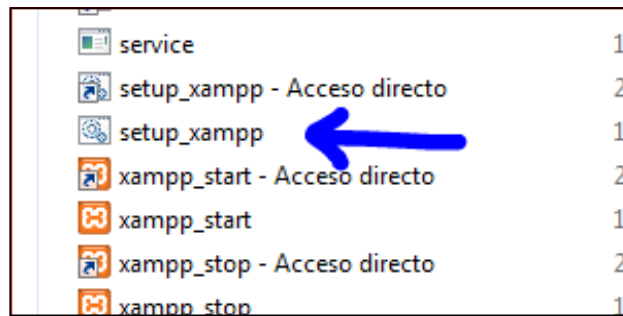
Hacemos clic en la opción de Extraer todo... y se nos muestra la siguiente ventana



Tenemos que tener cuidado para que la ruta de directorio no sea tan larga, que descomprimos la carpeta XAMPP sin mantener la ruta que tenía. Como resultado debe quedar la carpeta XAMPP dentro de la carpeta SERVWEB, tal y como se muestra en la figura



Hacemos doble clic en la carpeta XAMPP y hacemos doble clic en el fichero "setup_xampp.bat" para ejecutarlo



El resultado de la ejecución es el que se muestra en la siguiente ventana

```
#####
# ApacheFriends XAMPP setup win32 Version
#-----
# Copyright (c) 2002-2012 ApacheFriends 1.8.0
#-----
# Authors: Kay Vogelgesang <kvo@apachefriends.org>
#          Carsten Wiedmann <webmaster@wiedmann-online.de>
#####

Configure XAMPP with awk for 'Windows_NT'
Updating configuration files ... please wait ... DONE!

##### Have fun with ApacheFriends XAMPP! #####

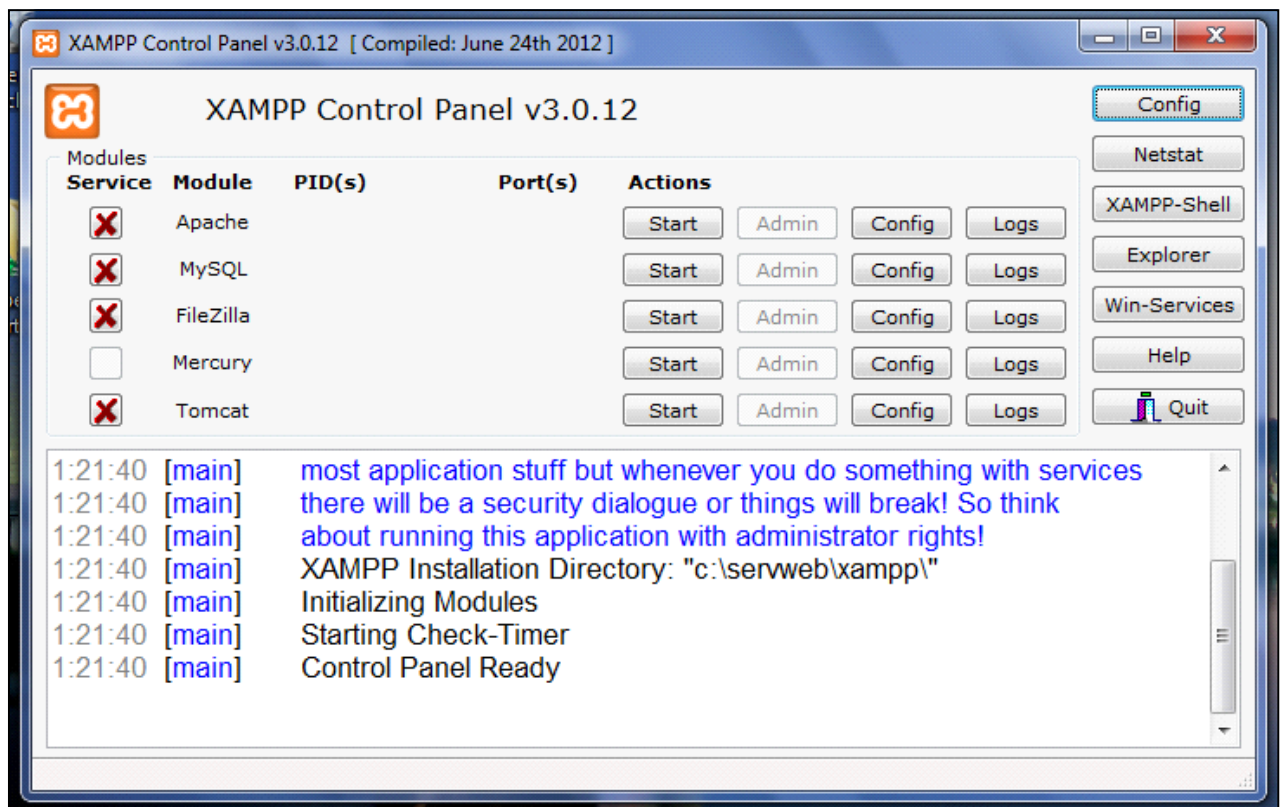
Presione una tecla para continuar . . .
```

Presionamos cualquier tecla para continuar

Una vez realizada la configuración de los ficheros de XAMPP, procedemos a ejecutar su panel de control haciendo doble clic en el fichero “xampp_control.exe”

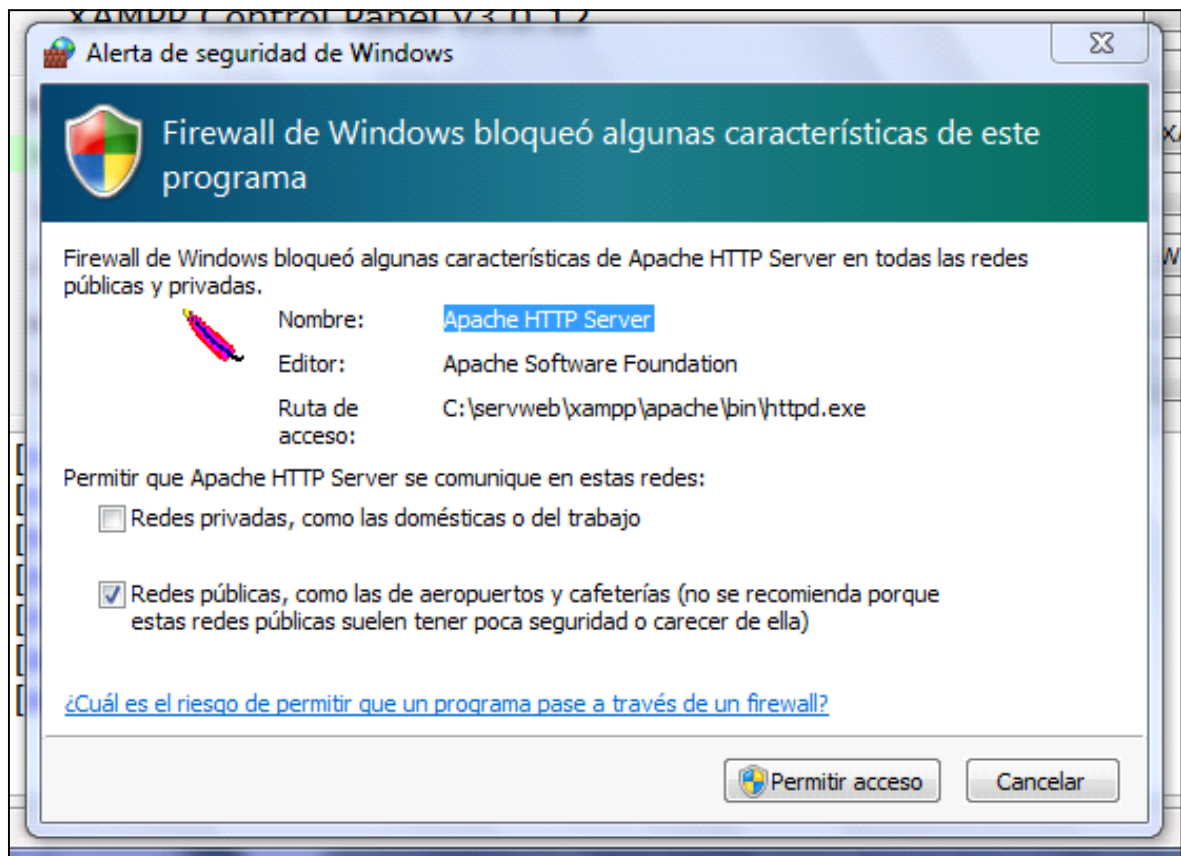


Nos sale esta ventana para escoger el idioma de presentación, en este caso escogemos inglés, y se muestra la ventana principal del panel de control de XAMPP.

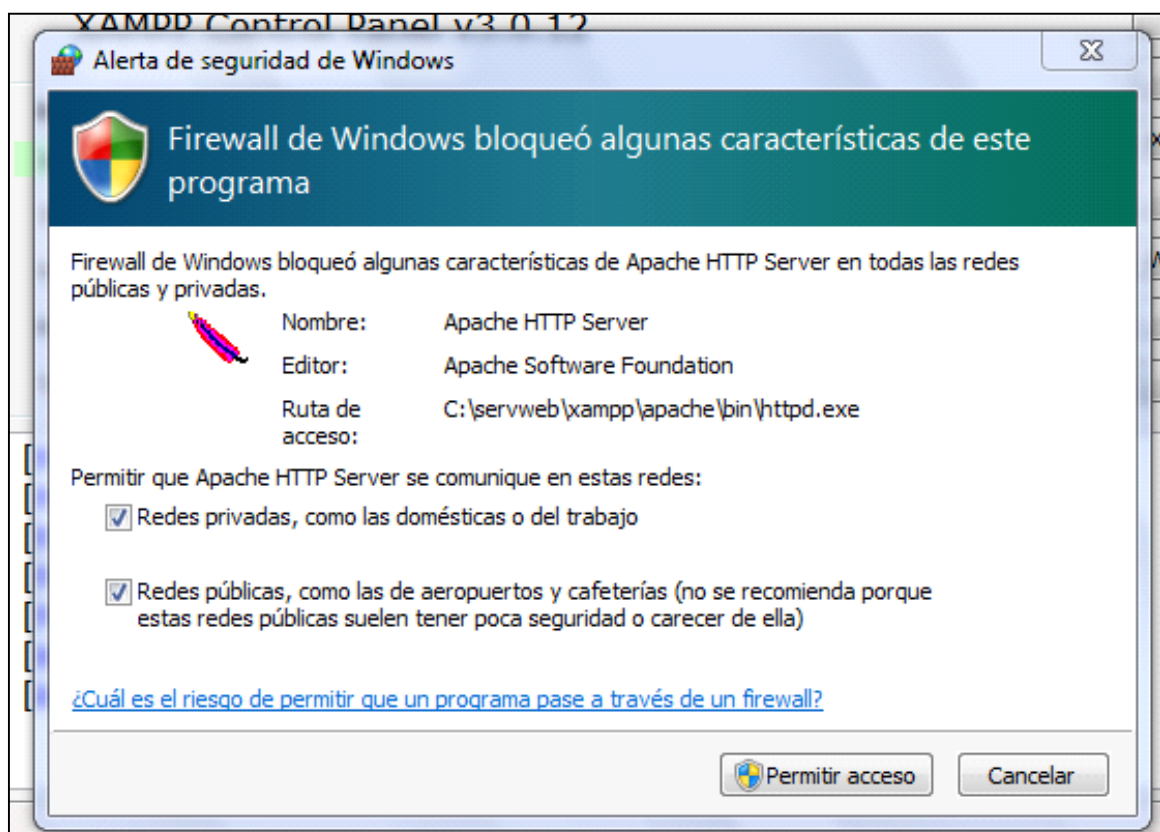


Vamos a iniciar ahora el servidor apache (servidor web). Para ello hacemos clic en el botón Start en la línea donde aparece el servicio de Apache

Se nos muestra una ventana del firewall de Windows que bloquea la salida el servidor apache y nos pide autorización para su ejecución



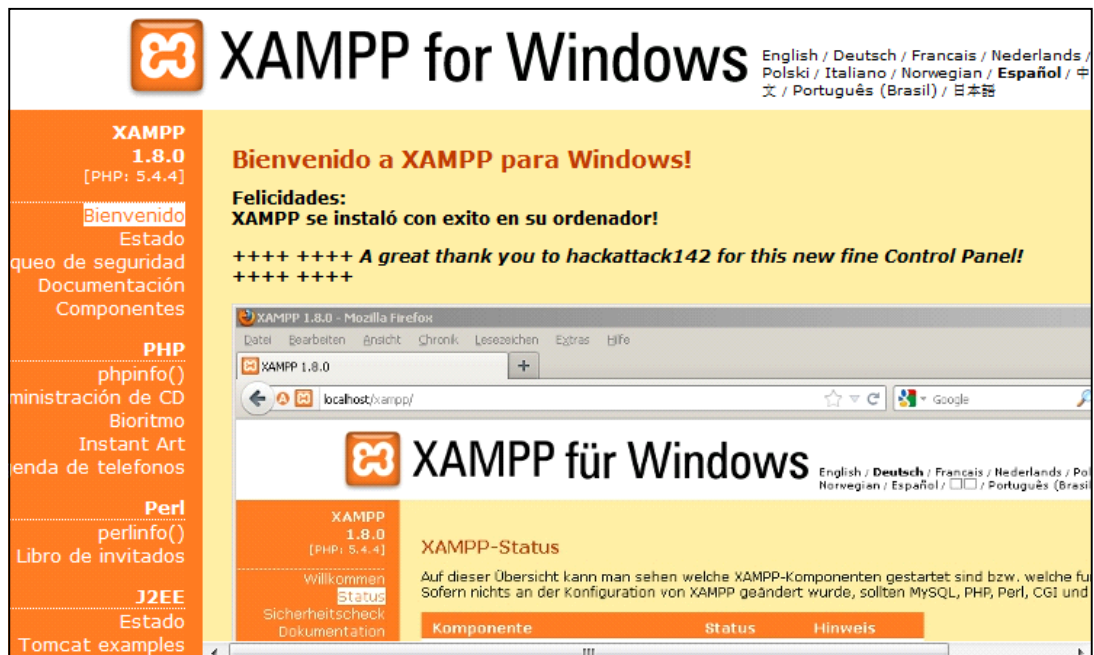
marcamos la opción de permitir a Apache se comunice en las redes privadas y le damos clic al botón de “Permitir acceso”.



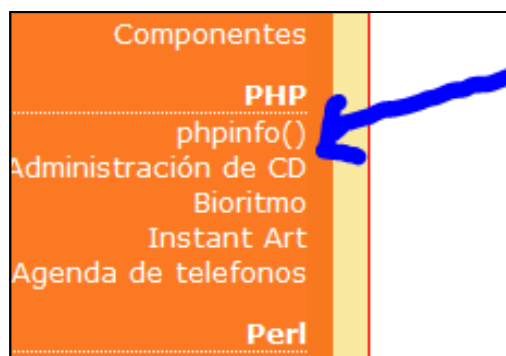
Y ya tenemos ejecutando el servidor web. Para probarlo abrimos el navegador de nuestra preferencia y como dirección ponemos “localhost”, el resultado debería ser el que se muestra en la siguiente ventana:



Escogemos el idioma Español




Para comprobar que PHP está funcionando hacemos clic en la opción de phpinfo(), en el panel izquierdo de la ventana en la sección de PHP



Si todo funciona correctamente debemos ver una ventana como la que se muestra a continuación:



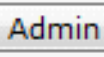

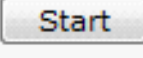



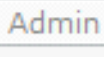
日本語

PHP Version 5.4.4

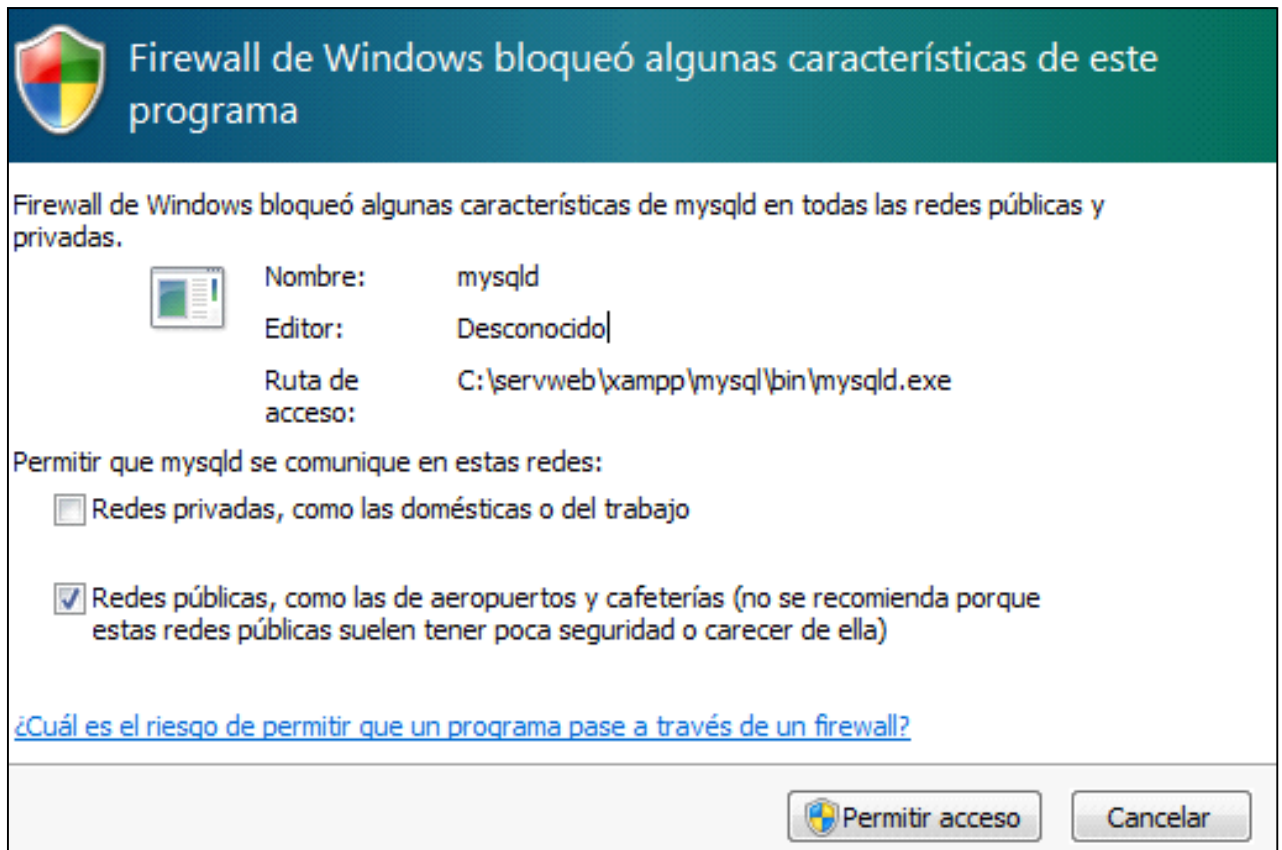


System	Windows NT WIN-K04FMO0PTAC 6.1 build 7601 (Windows 7 Enterprise Edition Service Pack 1) i586
Build Date	Jun 13 2012 21:17:57
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.4 Handler Apache Lounge
Virtual Directory Support	enabled

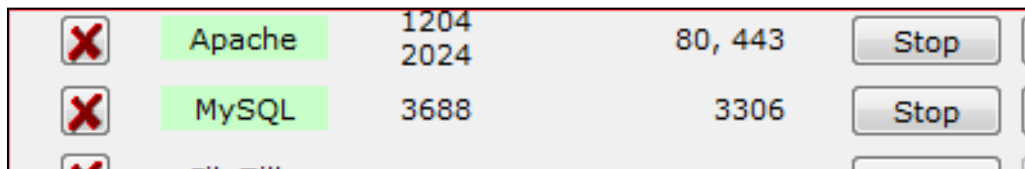
Vamos a iniciar el servicio de base de datos que será ofrecido por MySQL, para ello en el panel de control de XAMPP hacer clic en el botón Start de la línea correspondiente a MySQL.

Service	Module	Port(s)	Port(s)	Actions
	Apache	1204 2024	80, 443	 
	MySQL			 
	FileZilla			 

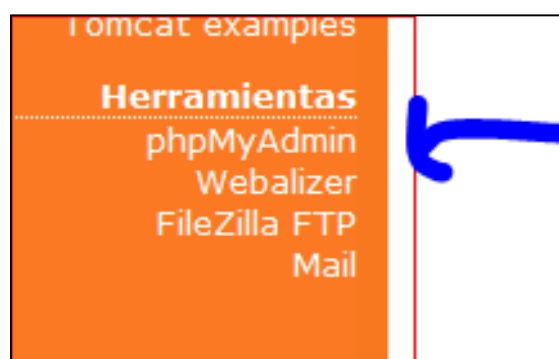
Al hacer clic en el botón nuevamente se presenta una ventana de la ejecución del firewall de Windows impidiendo la salida del servicio que estamos activando



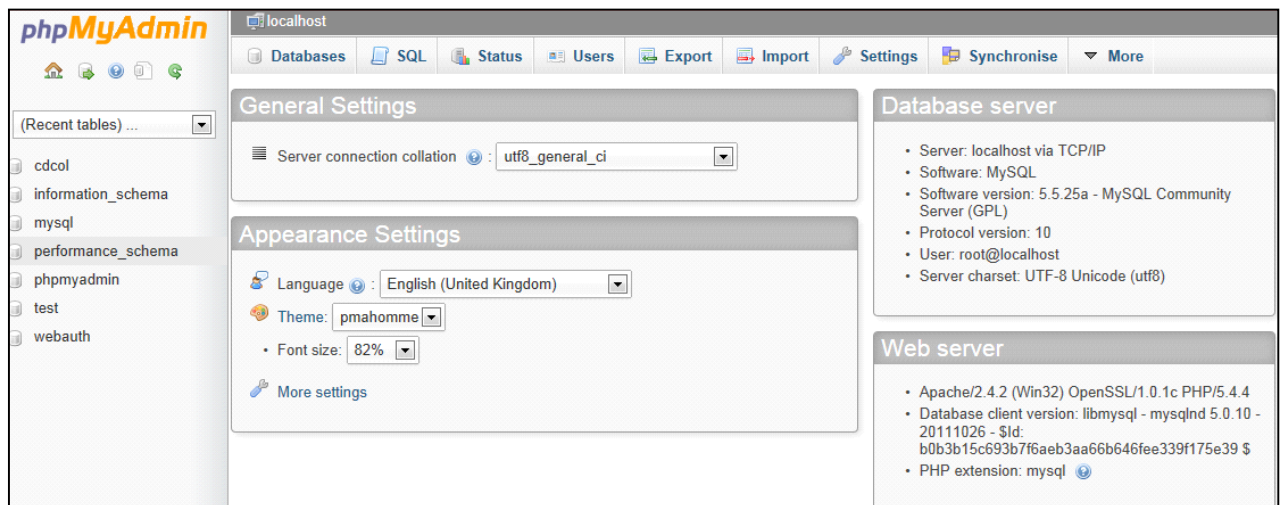
Al igual que antes, marcar la opción de que mysqld se comunice en las redes privadas, y hacer clic en el botón “Permitir acceso”.



Para comprobar que MySQL está ejecutándose, vamos a ejecutar phpMyAdmin para ver el contenido de las bases de datos. Para ello, en el panel izquierdo del navegador y bajo el apartado de “Herramientas”, se encuentra el enlace hacia phpMyAdmin, tal y como se muestra en la figura



hacemos clic en phpMyAdmin y se muestra



Procedemos ahora a configurarlo en español, para ello en la ventana

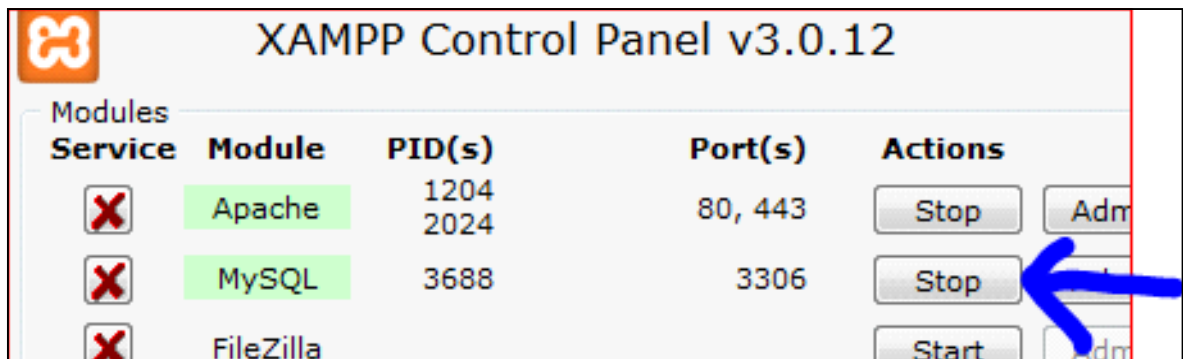


Cambiamos el lenguaje a español y si quisiéramos podemos cambiar la apariencia de las ventanas si escogemos otro tema.

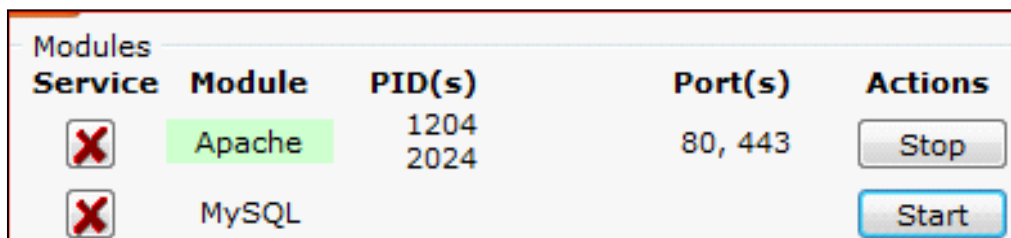


Ya hemos verificado que nuestro servidor XAMPP está funcionando correctamente.

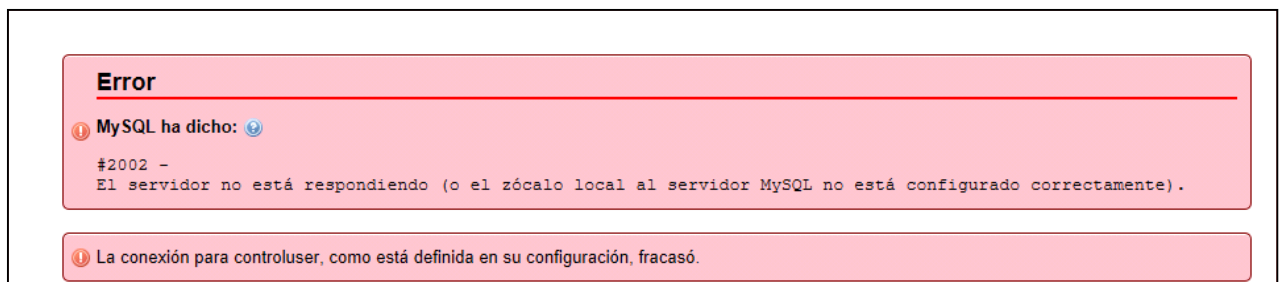
Para detener cualquiera de los servicios, debemos hacer clic en el botón “Stop” del servicio a detener.



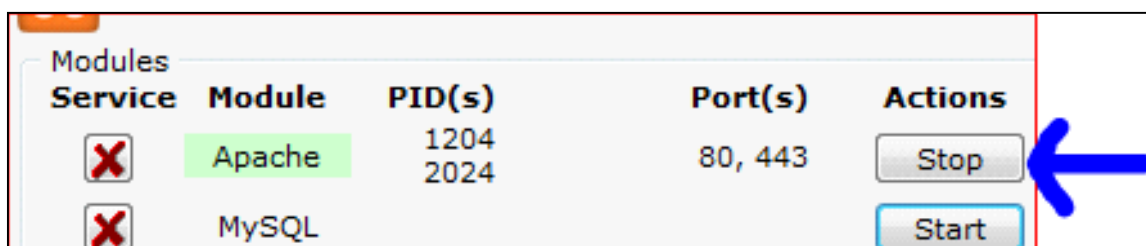
Para detener el servicio de MySQL hacemos clic en el botón Stop resaltado






Y al intentar ejecutar phpMyAdmin nuevamente

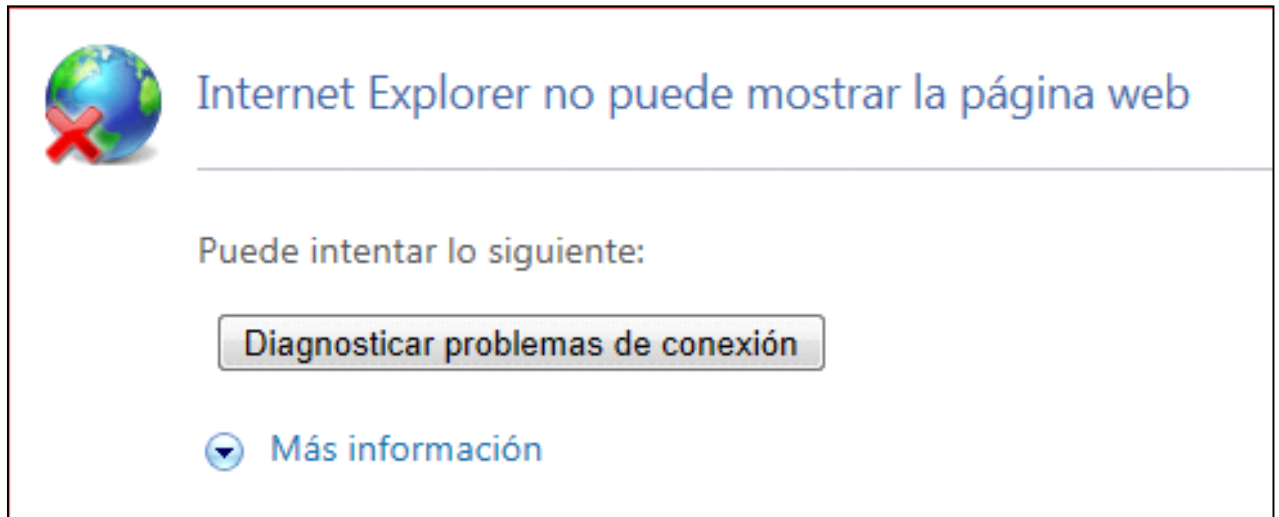


Otro mensaje de error nos aparece cuando intentamos acceder a localhost cuando el servidor web no está operativo.



Modules				
Service	Module	PID(s)	Port(s)	Actions
	Apache			Start
	MySQL			Start
	PHP			Start

Y luego se intenta navegar en localhost



03. Instalación de las herramientas de programación necesarias para trabajar con php.

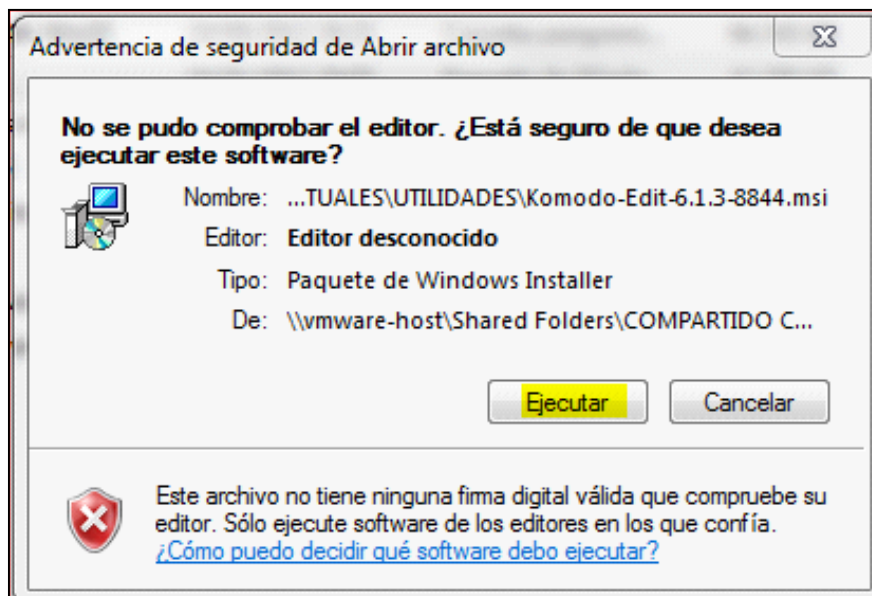
Una vez instalado y configurado nuestro entorno de trabajo, debemos instalar las herramientas que nos permitan crear nuestros programas en php. Estamos hablando de los editores de programación y de los debugger (seguidores de ejecución).

Nosotros vamos a instalar dos editores (ambos gratuitos) para la programación en php, estos son Komodo Edit y Eclipse PDT para PHP.

03.1 Instalación de Komodo Edit

Komodo Edit es un editor de lenguaje de programación básico, pero muy bueno. Con el se puede trabajar en muchos lenguajes de programación. Una vez descargado el fichero de Komodo Edit procedemos a su instalación haciendo doble clic sobre el

VMWARE FUSION 5	23
7z920	21
eclipse-php-3.0.2.v20120611144-Mac.tar.gz	19
eclipse-php-3.0.2.v20120611144-Win32	19
Komodo-Edit-6.1.3-8844	04
php_xdebug-2.2.1-5.4-vc9-nts-x86_64.dll	19
php-5.4.7-nts-Win32-VC9-x86	23
php-debug-pack-5.4.7-nts-Win32-VC9-x86	23
xampp-win32-1.8.0-VC9	20
ZendDebugger-20100729-darwin9.5-x86_...	19
ZendDebugger-20110410-cygwin_nt-i386	19



Hacemos clic en Ejecutar



ActiveState Komodo Edit 6.1.3



ActiveState



Komodo[®] Edit

www.activestate.com

**Welcome to the ActiveState Komodo Edit
6.1.3 Setup Wizard**

The Setup Wizard will install ActiveState Komodo Edit 6.1.3 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Back

Next

Cancel



End-User License Agreement

Please read the following license agreement carefully

ActiveState

MOZILLA PUBLIC LICENSE Version 1.1

1. Definitions.

1.0.1. "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.

1.1. "Contributor" means each entity that creates or contributes to

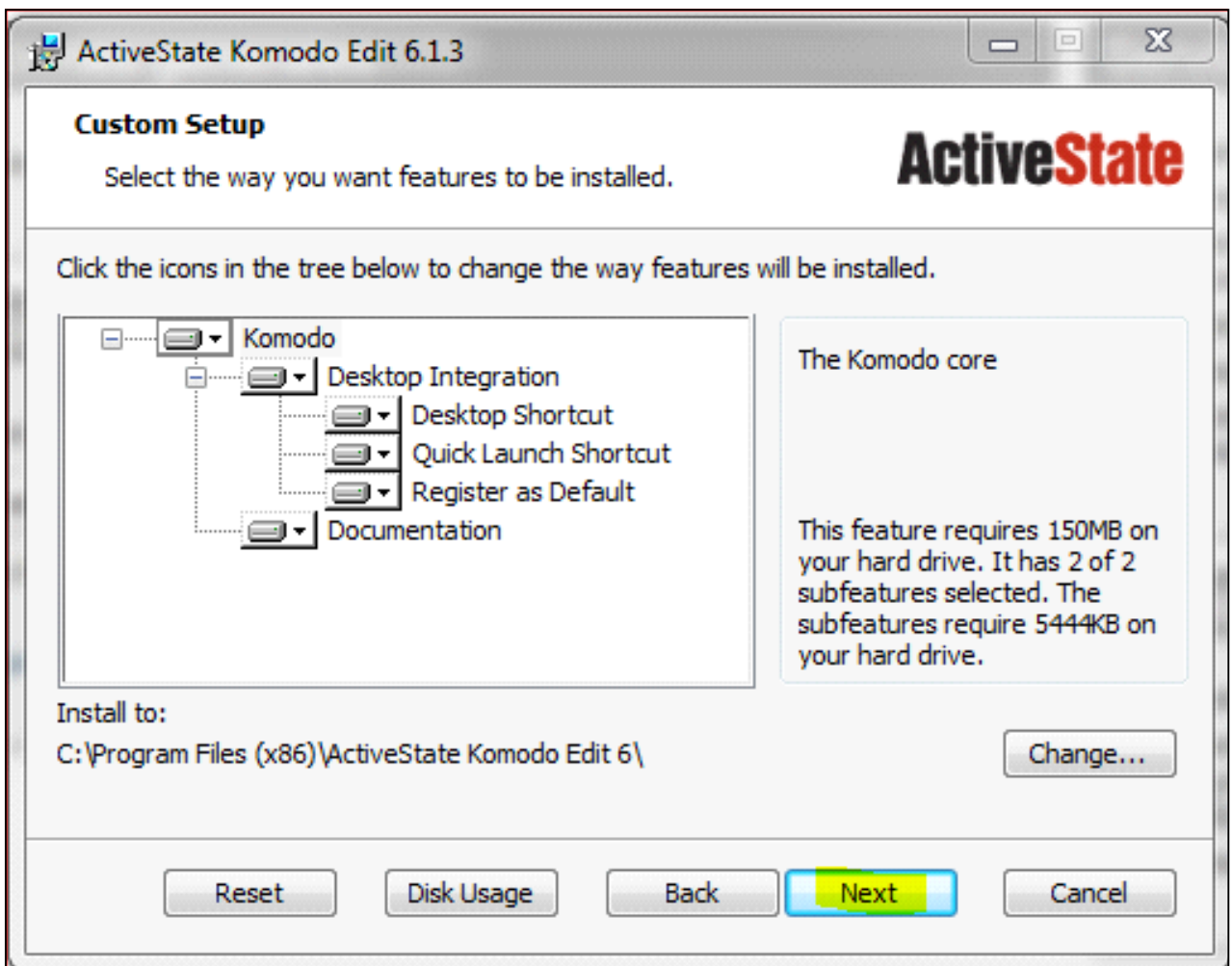


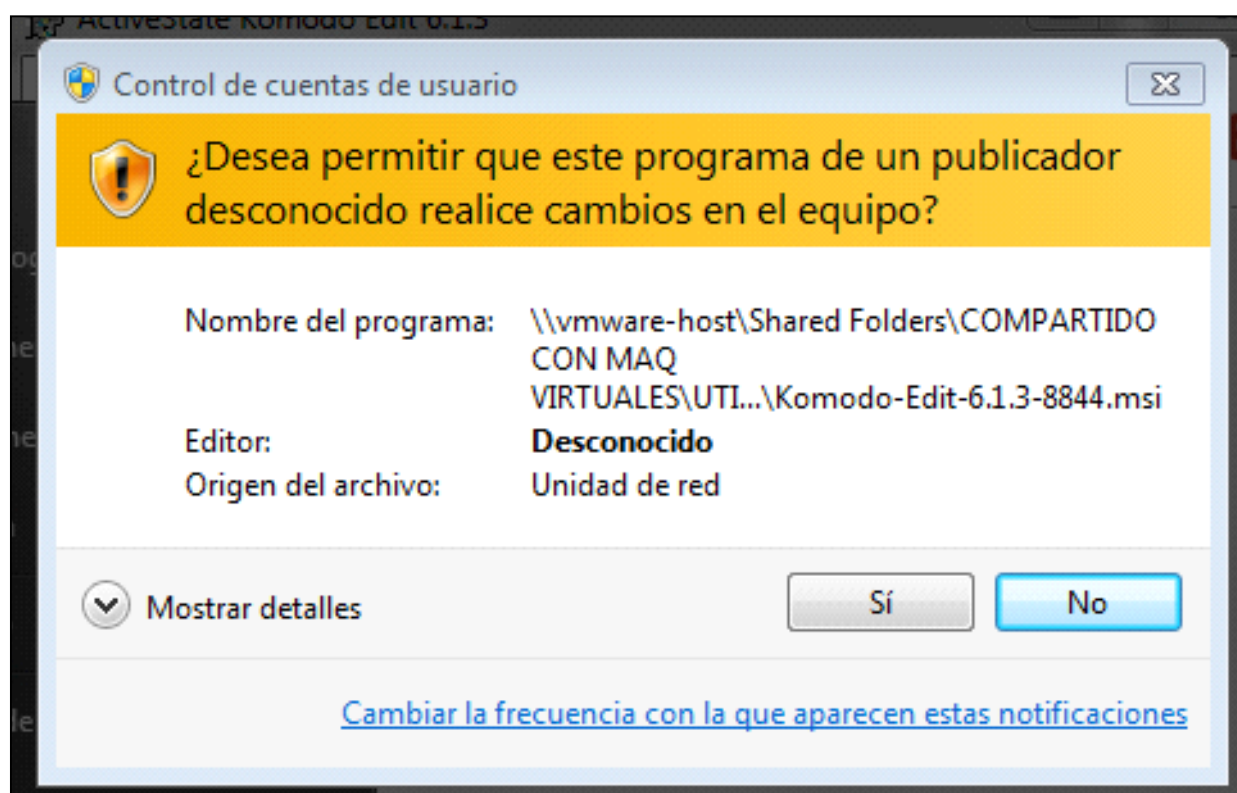
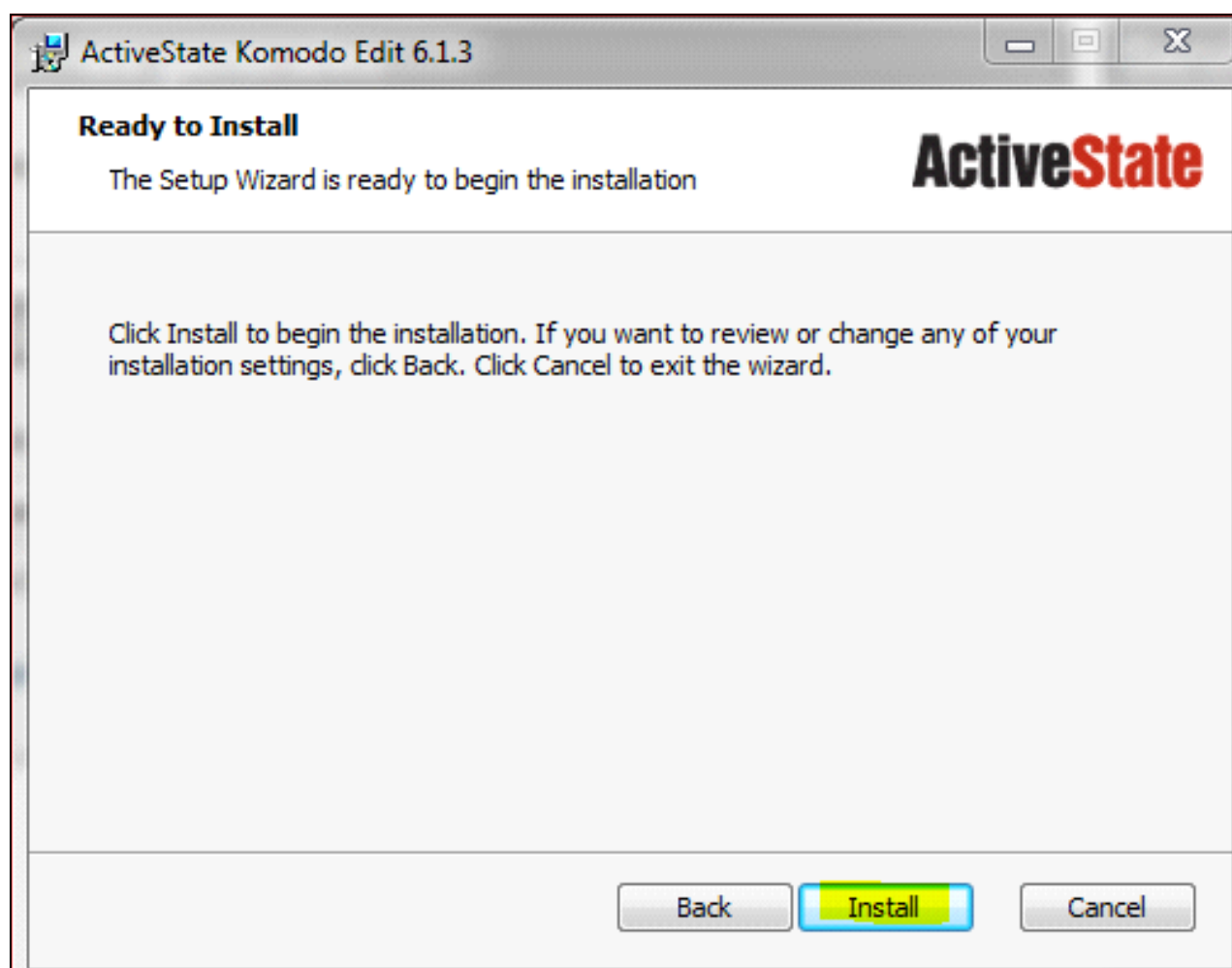
I accept the terms in the License Agreement

Back

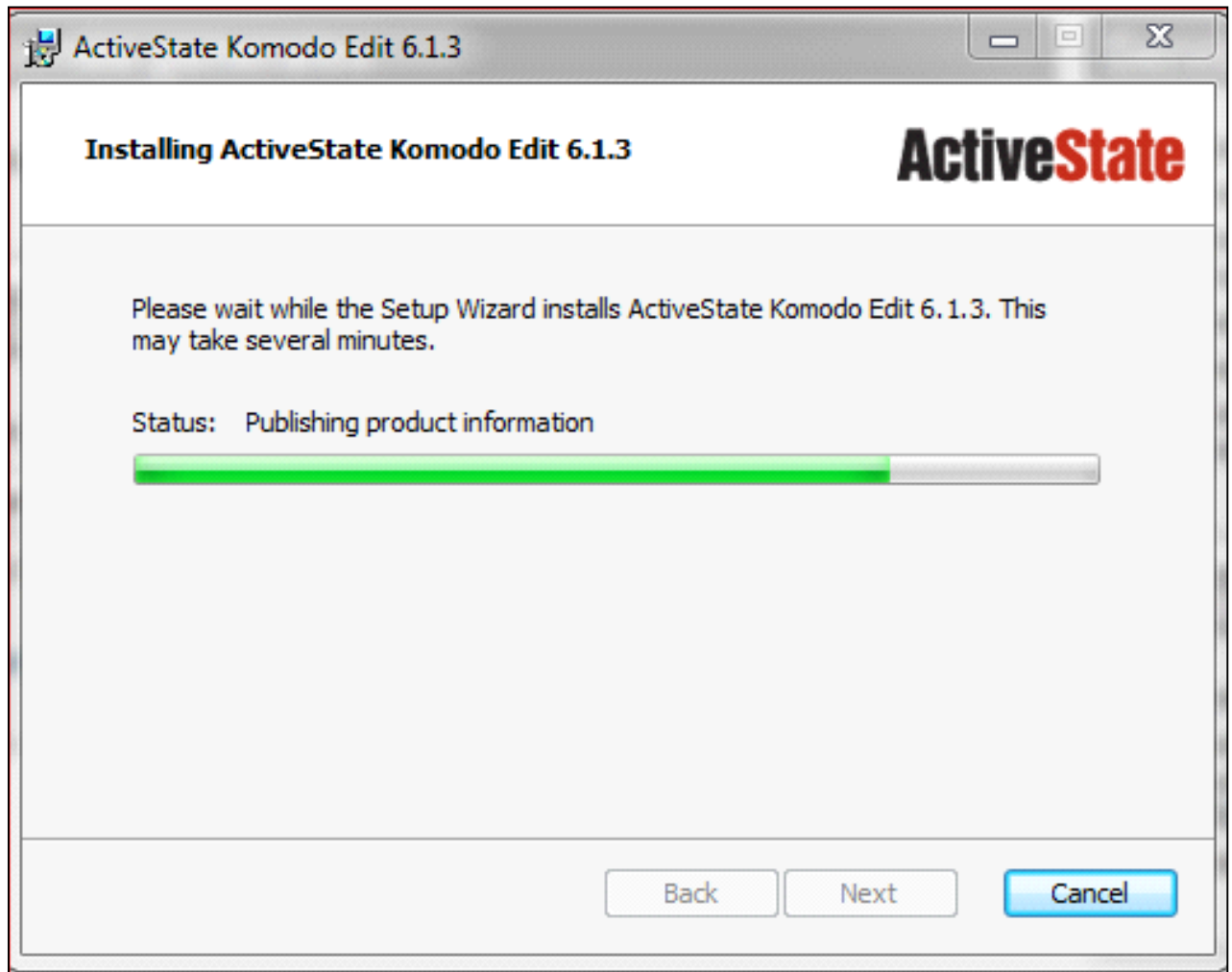
Next

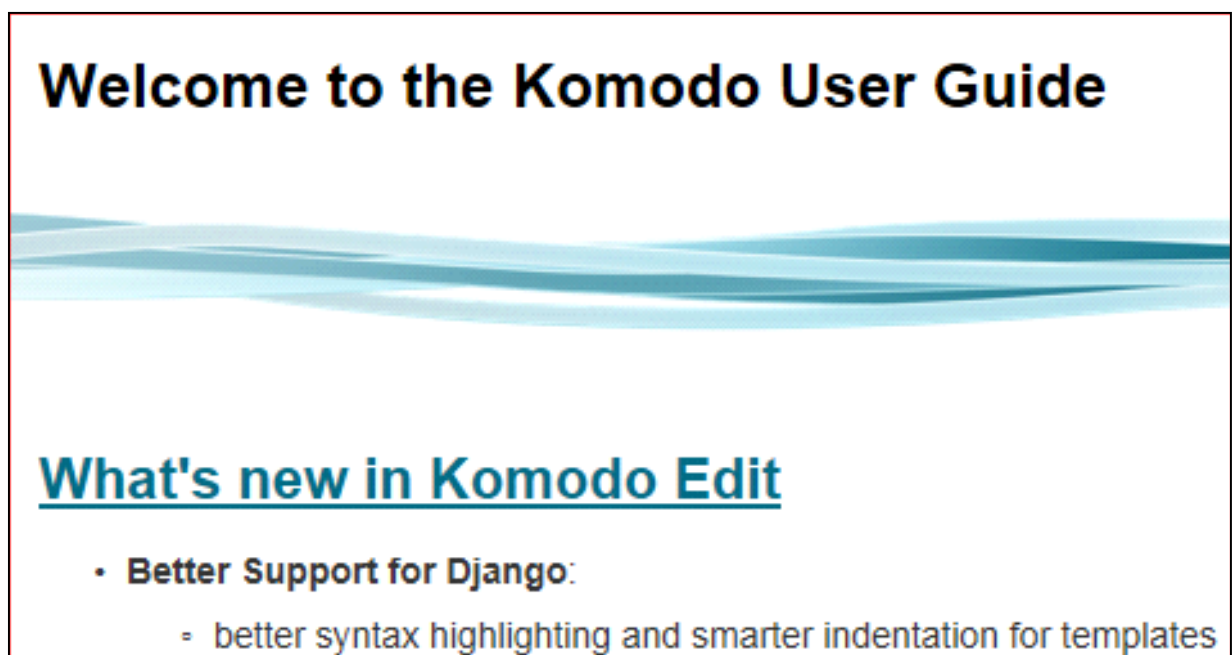
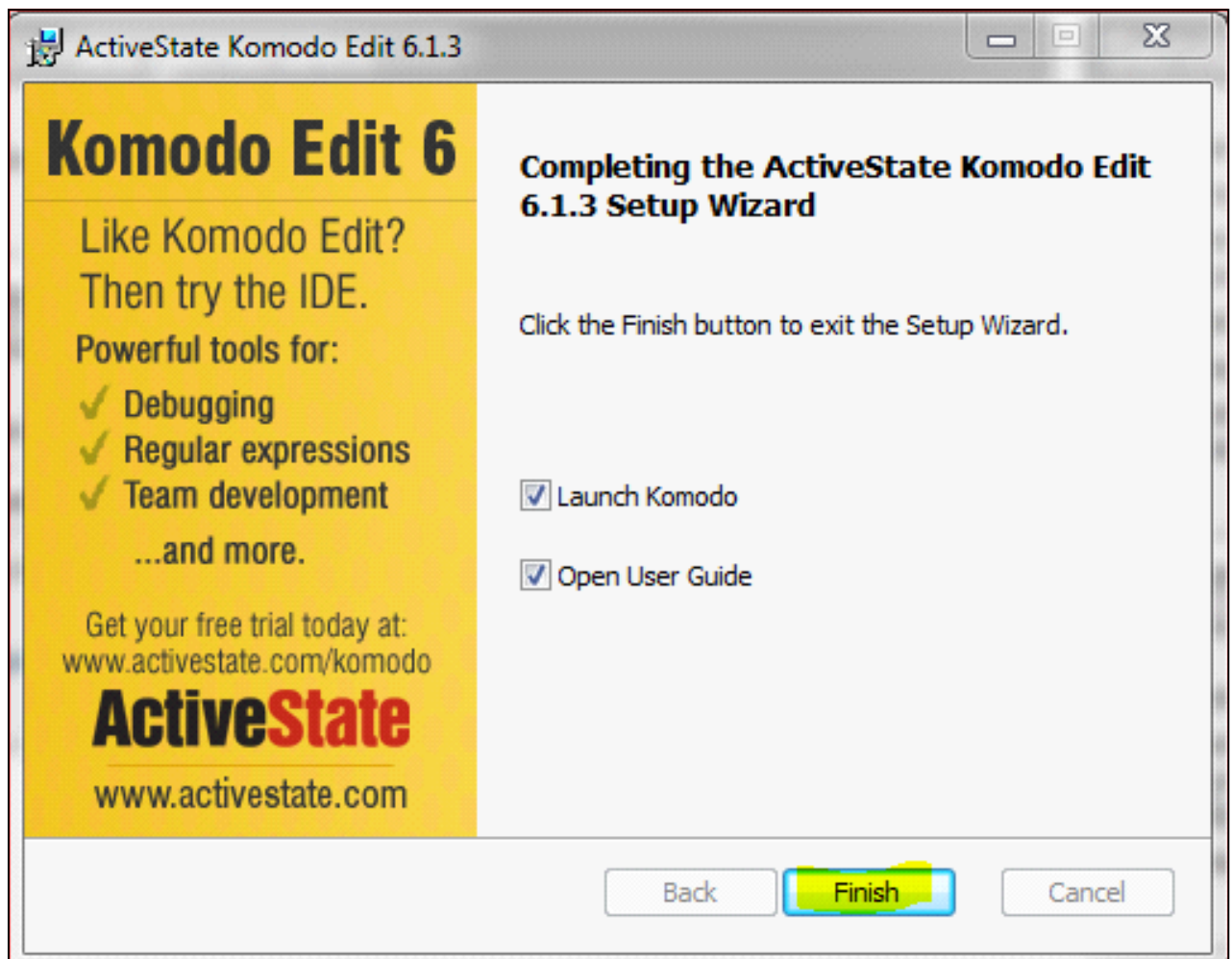
Cancel

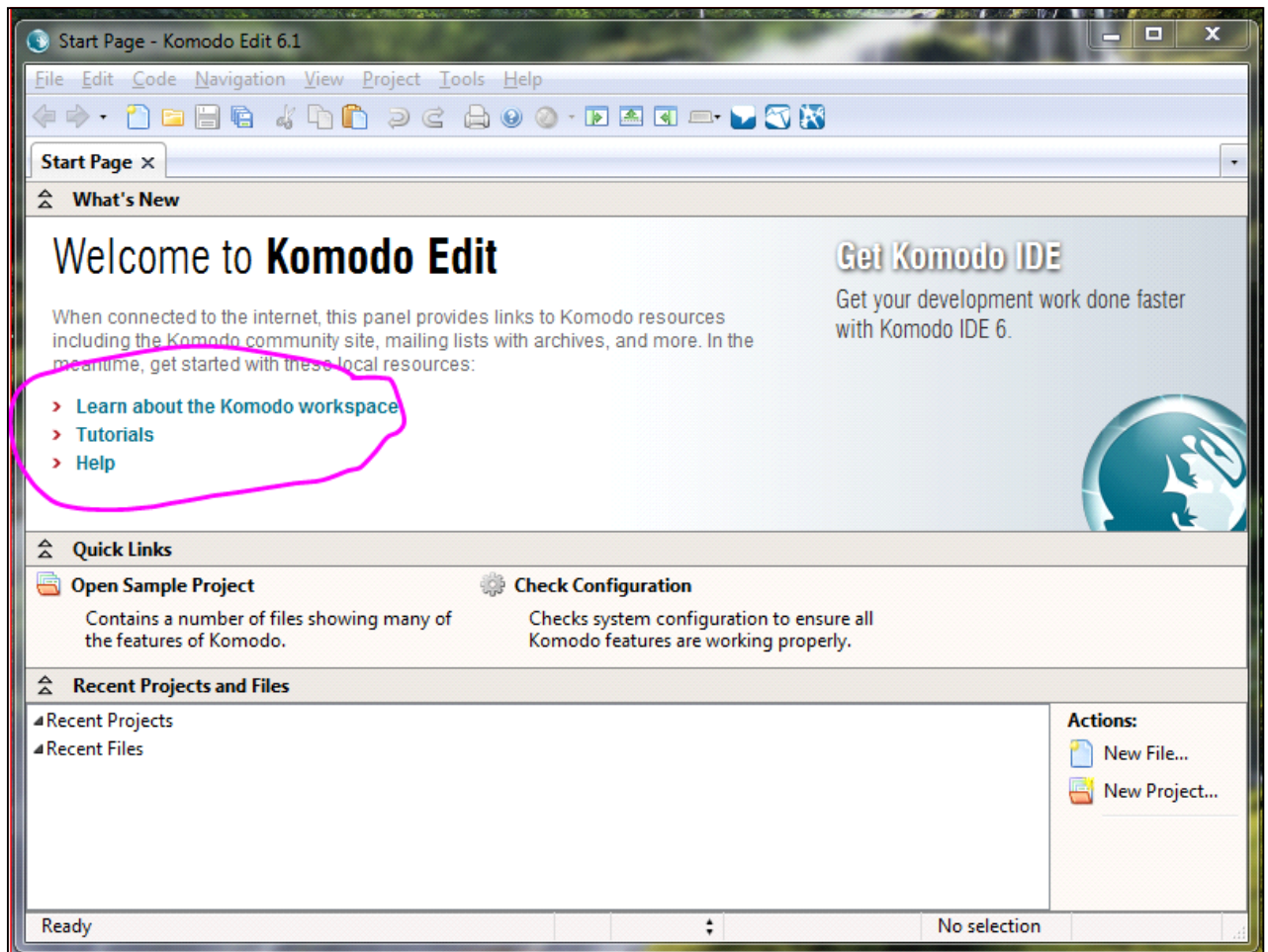




Hacemos clic en el botón Si







03.2 Instalación de Eclipse PDT

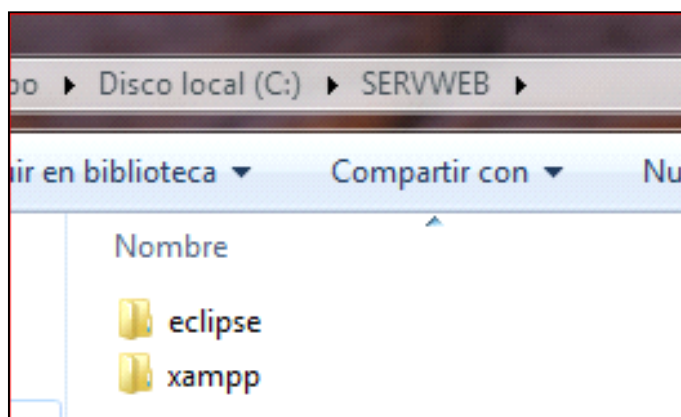
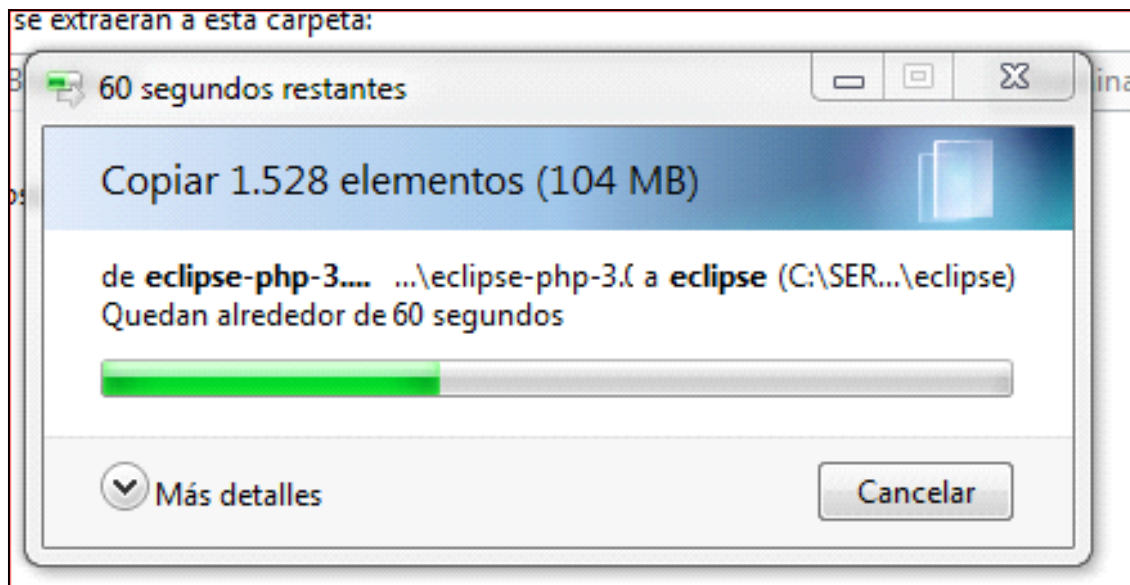
Una vez descargado el fichero de instalación de Eclipse PDT, creamos una carpeta de nombre eclipse en c:\SERVWEB. Extraemos el contenido del fichero descargado a la carpeta creada.

Seleccionar un destino y extraer archivos

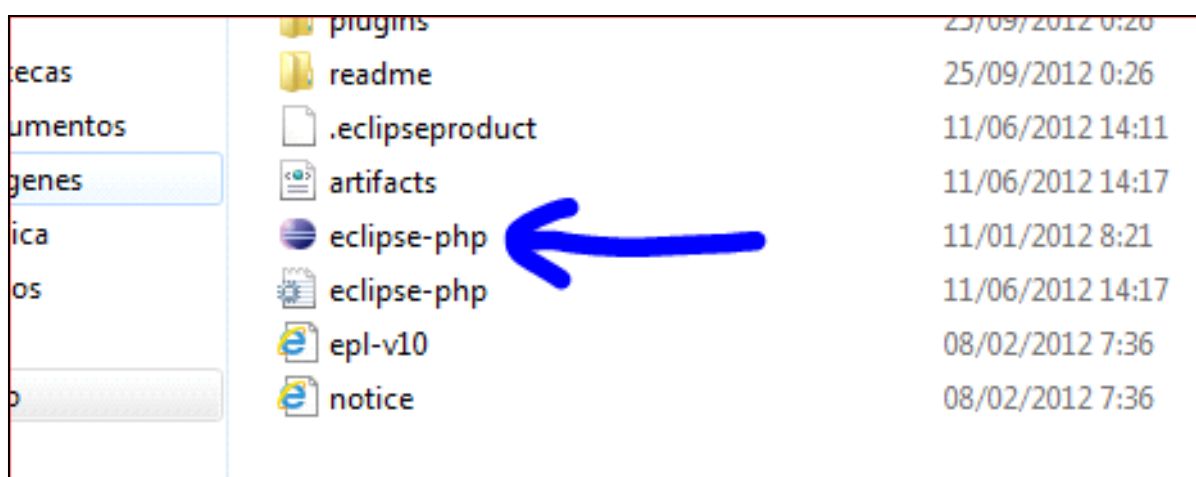
Los archivos se extraerán a esta carpeta:

Examinar...

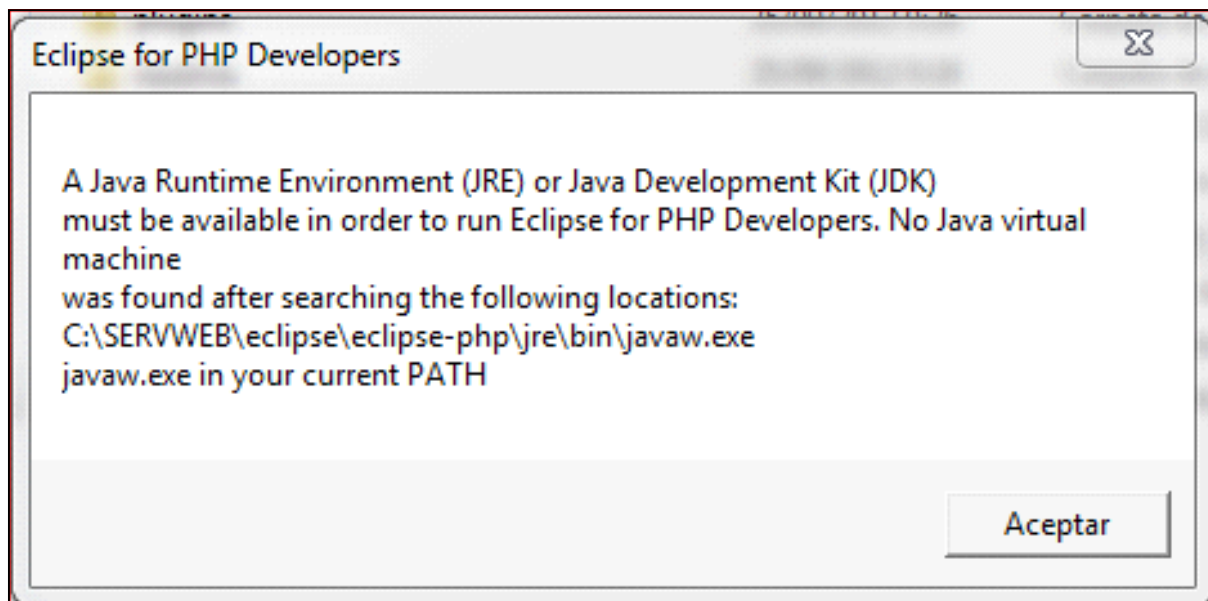
☒ Mostrar los archivos extraídos al completar



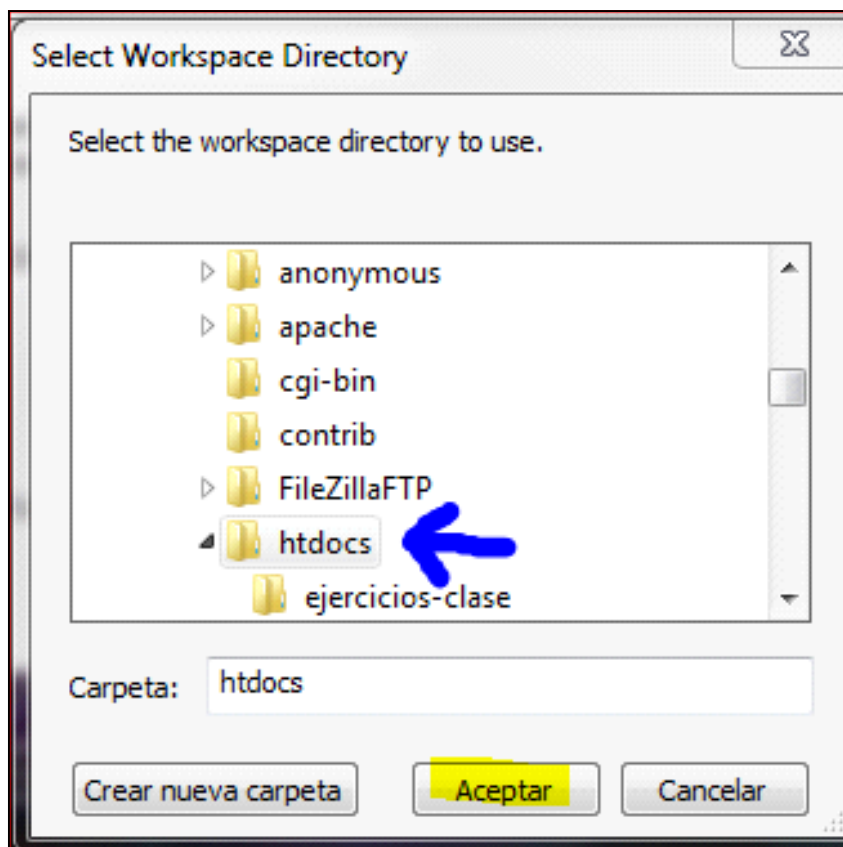
Hacemos doble clic en la carpeta



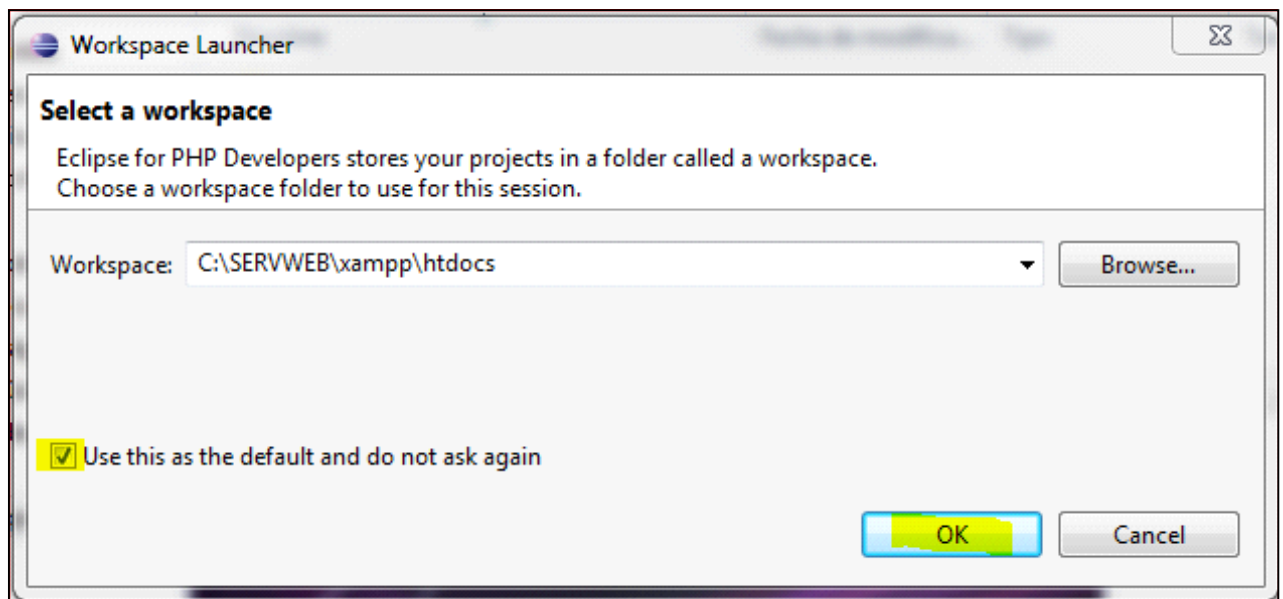
Y ahora hacemos doble clic en el fichero eclipse-php. Si no tenemos instalada una máquina virtual de Java se produce el siguiente error.



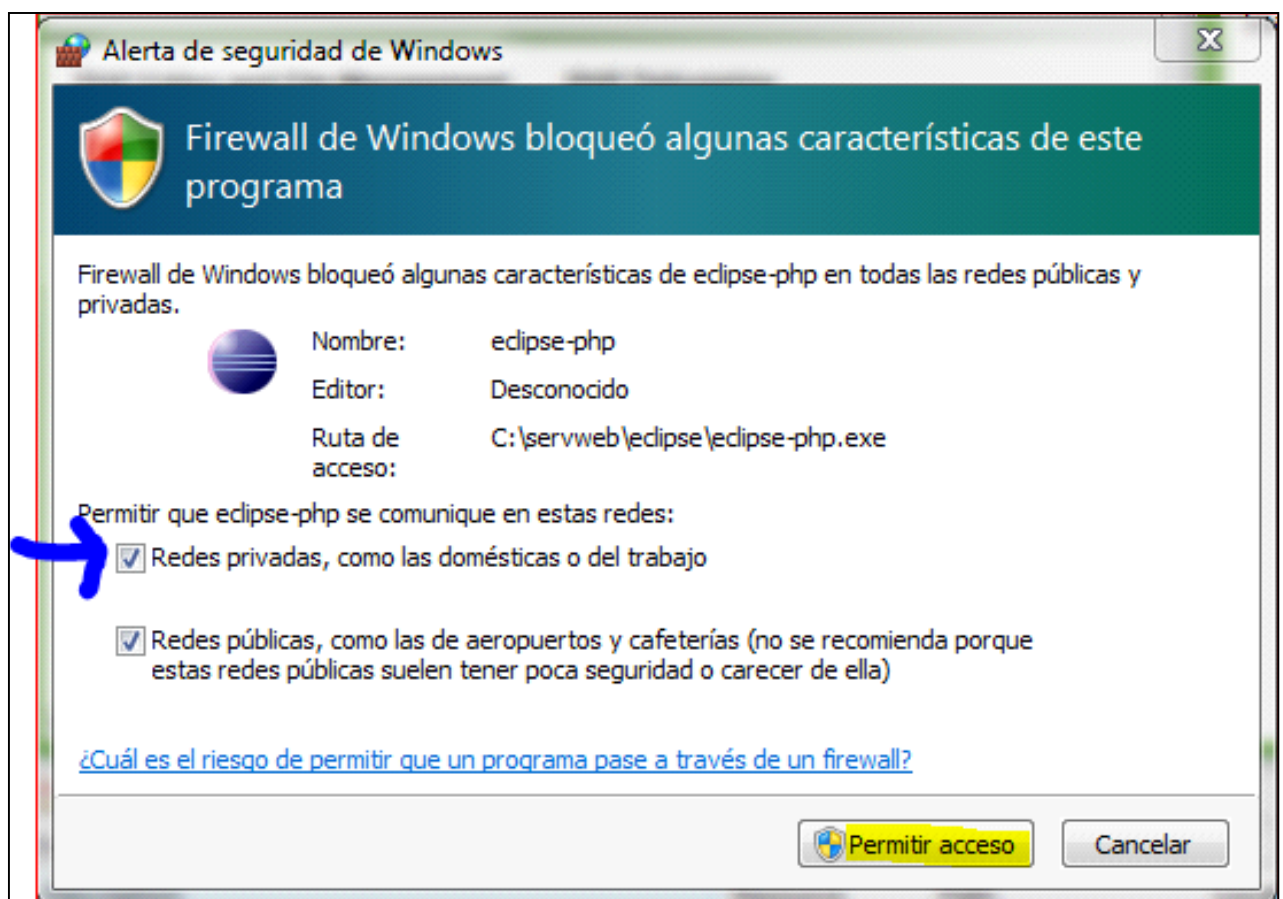
Procedemos a instalar el Java Runtime Environment de java (JRE) y hacemos nuevamente clic en el fichero eclipse-php.



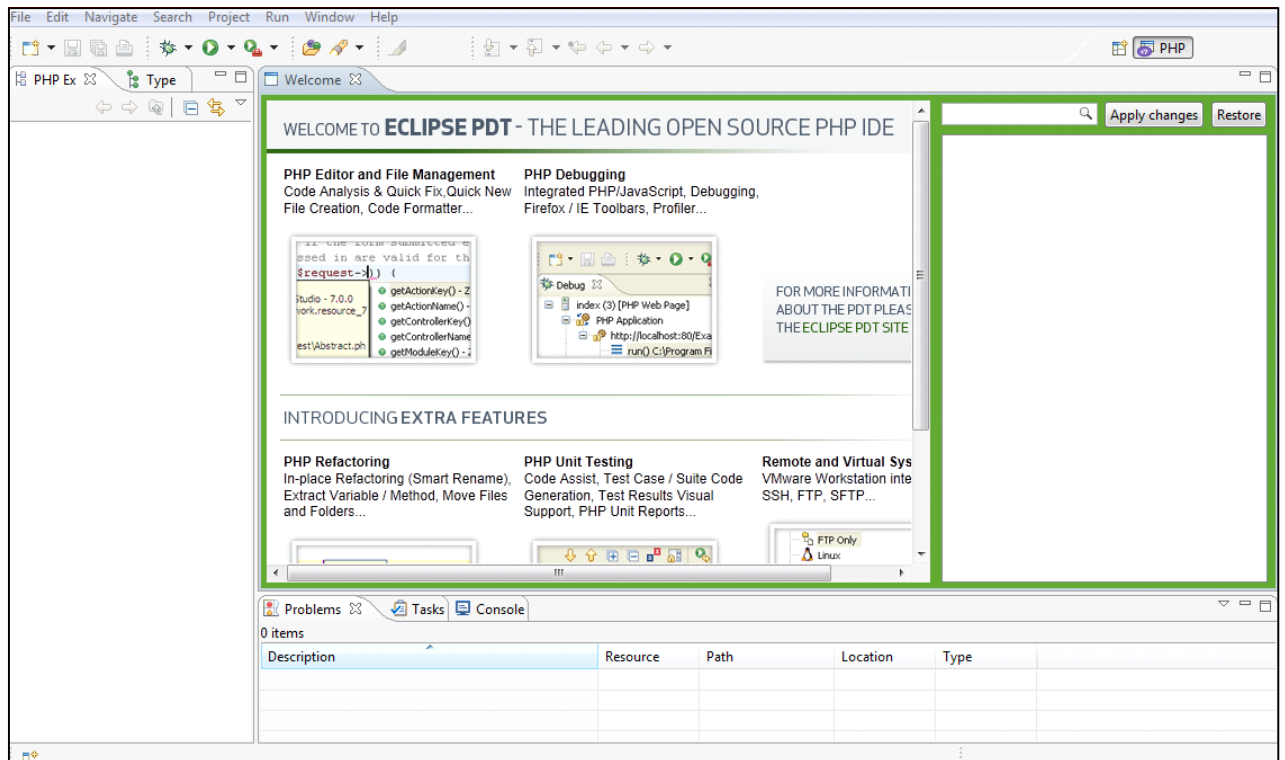
Hacemos clic en Aceptar



Clic en el botón OK



Creamos una regla en el firewall de Windows para que permita se comunique en red. Hacemos clic en el botón "Permitir acceso".



Crear un proyecto

New PHP Project

Create a PHP Project
Create a PHP project in the workspace or in an external location.

Project name: **proyecto-prueba**

Contents

- ☒ Create new project in workspace
- ☐ Create project at existing location (from existing source)

Directory: C:\SERVWEB\xampp\htdocs\proyecto-prueba [Browse...](#)

PHP Version

- ☒ Use default PHP settings
- ☐ Use project specific settings: PHP Version: **PHP 5.3**

Project Layout

- ☒ Use project as source folder
- ☐ Create separate folders for source files and public resources [Configure default...](#)

JavaScript Support

- ☐ Enable JavaScript support for this project

[?](#) [< Back](#) [Next >](#) **Finish** [Cancel](#)

04. Instalación de las herramientas de programación necesarias para trabajar con php.

Una vez realizado la instalación de las que serán nuestras herramientas de trabajo comencemos a programar en un lenguaje de script de servidor (PHP).

Como se dijo anteriormente el código de PHP está incrustado dentro del código de una página web, es decir, dentro del código HTML de la página web. Recordemos ahora la sintaxis de una página web escrita en este caso en XHTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es" >
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Bienvenido a esta página!</title>
</head>
<body>
<h1>Esta es una página básica de xhtml!</h1>
<br/>
<p>Para mostrar la estructura de un página en xhtml</p>
</body>
</html>
```

Fichero: 01-estructura.html

Para agregar código PHP a una página HTML se debe encerrar entre las etiquetas `<?php` y `?>`, todo el código que se ponga entre ellos será interpretado como código PHP. Existen otras formas de etiquetas para encerrar el código PHP pero ésta es la más estandarizada.

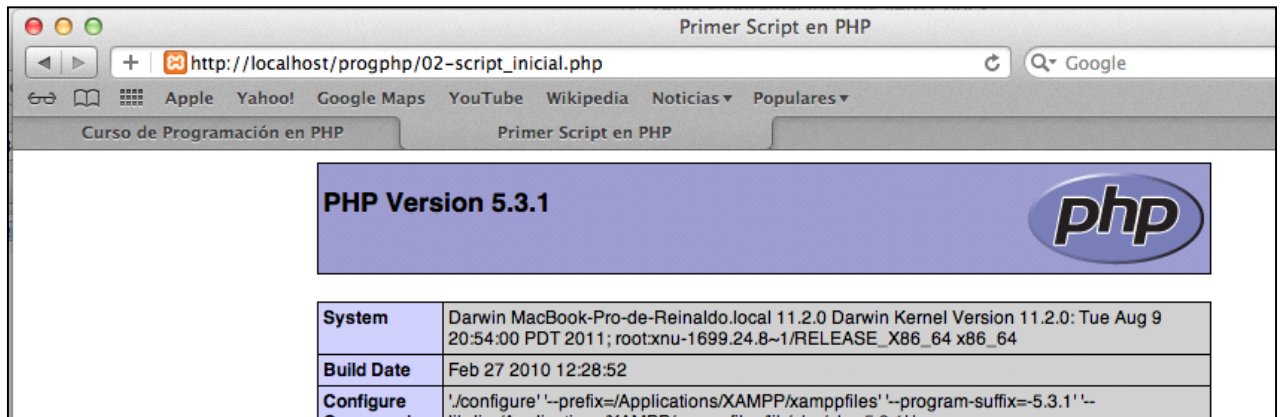
Ejecutemos el siguiente código, donde hemos introducido un script en PHP para testear si nuestro servidor está ejecutándose correctamente y ver las opciones instaladas (librerías) instaladas.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Primer Script en PHP</title>
</head>
<body>
<?php
    |phpinfo();
?>
</body>
</html>
```

Fichero: 02-primer_script_php.php

Un fichero que contenga código PHP debe tener extensión .php. Para ejecutar el código, debe realizarse a través de un navegador y se debe estar ejecutando un servidor web que pueda interpretar el código de PHP.

Un típico error al intentar ejecutar un fichero de php es hacer doble clic sobre él (eso no funciona).



Vemos que nuestra instrucción se ha ejecutado correctamente.

DESCRIPCIÓN DEL LENGUAJE PHP

Toda instrucción en PHP termina en ';'.

Para escribir comentarios en nuestro código, se puede hacer de dos maneras, ambas se muestran a continuación.


```

<?php

// siempre es bueno comentar los programas
// ya que nos ayudará a recordar lo que hemos hecho
// a la hora de tener que modificarlos posteriormente

// esto es un comentario de una línea
// esto es un comentario en una segunda línea

$x = 1; // el comentario puede comenzar al terminar la instrucción
        // todo lo que aparezca después del // no se tendrá en cuenta

// existe otra manera de poner comentarios de varias líneas
// en un programa de php. Esta es igual a la de muchos
// lenguajes de programación

// este tipo de comentario comienza con un /* y termina con */

// un ejemplo

/*
    esto son comentarios
    que pueden abarcar varias líneas
    hasta que no se escriba los caracteres
    de finalización
    Este tipo de comentario no se puede anidar.
*/

?>

```

Fichero: 03-comentarios en php.php

Recordamos además como se realizan los comentarios en HTML

```

?>

<!-- Esto es un comentario en html -->

```

PHP es un lenguaje que no distingue de mayúscula o minúsculas a nivel de comandos, pero si lo distingue a nivel de las variables y funciones que vayamos a crear.

ESCRIBIR EN PHP (Enviar datos al navegador)

Para escribir en PHP se usa la instrucción “echo” o la instrucción “print”, la manera de usar es la que se muestra a continuación.

```
print 'frase a escribir'
print "frase a escribir"
echo 'frase a escribir'
echo "frase a escribir"
```

El texto literal que se quiera escribir puede ponerse entre comillas simples o dobles. La única diferencia es que en las comillas dobles las variables (son sustituidas por sus valores) y los caracteres especiales (son interpretados) mientras que en las comillas simples se toma como literales. Todas las etiquetas de HTML para dar formato al texto son interpretadas por “print” pero no por “echo”.

Abramos con el editor el fichero 04-script_escribir.php

```
<?php

    $i = 0;

    echo "Hola ";
    print "Hola otra vez <br />";
    echo 'Hola ';
    print 'Hola otra vez';
    echo "Usando comillas dobles el valor de i es $i <br />";
    print 'Usando comillas simples el valor de i es $i <br />';
    echo 'para escribir comillas "dobles" <br />';
    print "para escribir comillas \"dobles\" <br />";
    print "para saltar la línea uso <br />";
    echo "esto está en otra línea"
```

Fichero: 04-script_escribir.php

Variables en PHP

Una variable es un área de memoria a la que se le asigna un nombre identificador y donde podemos guardar valores que puedan variar a lo largo del programa (de ahí su nombre de variable).

En PHP el nombre de las variables debe comenzar con el signo '\$' y el segundo carácter debe ser una letra o un guión bajo. No hay límite en la longitud del nombre. Y se diferencia mayúsculas de minúsculas.

No hace falta que las variables sean declaradas antes de ser utilizada, ni siquiera inicializarlas. A las variables a las que no se le da un valor inicial, PHP se lo asigna. Además, las variables no son de un tipo de dato determinado, es decir en un momento dado pueden ser un valor entero y más tarde una cadena de caracteres.

La asignación de una variable se realiza por valor lo que significa que el resultado de la expresión que está a la derecha del signo igual se copia (asigna) en la dirección de memoria de la variable que está a la izquierda del signo igual.

```
// asignación por valor
echo 'Asignación por valor';

$var1 = "Hola";
$var2 = 3;
$var3 = $var2;

print "$var1, $var2, $var3 <br />";
```

Fichero: 05-script_variables.php (Primera parte)

También es posible asignar valores por referencia (&) que significa que la dirección de la variable que está a la derecha del signo igual se asigna a la variable de la izquierda. En este caso sólo lo que se tiene es una variable con dos nombres.

```
// asignación por referencia
echo '<p>Asignación por referencia</p>';

$Cadena = "Tipo de dato string";
$Ref = &$Cadena; // en $Ref se guarda la dirección de $Cadena se usa &
echo "El valor de Cadena es = $Ref<br />";
echo "El valor de Ref es = $Cadena<br />";
$Cadena = "Asignamos otra cadena a la variable Cadena";
echo "El valor de Ref es = $Ref<br />";
$Ref = "Ahora asignamos otro valor a la cadena Ref";
print "El valor de Cadena es = $Cadena<br />";
```

Fichero: 05-script_variables.php (Parte final)

Ver fichero 06-mas-variables.php

REPORTE DE ERRORES EN PHP

Para que PHP nos informe sobre las variables que no hayamos inicializado debemos utilizar la sentencia o instrucción `error_reporting(E_ALL)`.

Para que los mensajes se muestren, en `php.ini` la variable `display_errors` debe estar en On

Si no tenemos acceso al fichero de configuración `php.ini` una forma de activar la variable `display_errors` en nuestro código es mediante la instrucción

```
ini_set("display_errors", 1);
```

Para su uso veamos el siguiente ejemplo:

```
<?php

//error_reporting(E_ALL);
//ini_set("display_errors", 1);

// include("file_with_errors.php"); si queremos incluir el codigo de un fichero

print "Probamos como funciona la correcci&oacute;n de errores <br />";
$a = $b + 3;
echo "El valor de la variable a es $a";

?>
```

Fichero: 07-script_reporta_errores.php

Que al ejecutarse lo hace correctamente. Pero no muestra el error de que la variable `$b` no tiene asignado ningún valor.

Al ejecutar este código la ventana del navegador sólo muestra el mensaje enviado por el comando `print`, modifiquemos el código para ahora agregar la instrucción `error_reporting`.

```
<?php

error_reporting(E_ALL);
ini_set("display_errors", 1);

// include("file_with_errors.php"); si queremos incluir el codigo de un fichero

print "Probamos como funciona la correcci&oacute;n de errores <br />";
$a = $b + 3;
echo "El valor de la variable a es $a";

?>
```

Fichero: 07-script_reporta_errores.php

y al ejecutarlo

TIPOS DE DATOS EN PHP

En PHP existen estos tipos de datos:

- **Integers:** son los números enteros
`$var = 20;`
- **Double:** son los números de coma flotante (decimales)
`$var = 250,89;` ó `$var = 25089e-2;`
- **String:** son las cadenas de caracteres
`$var = "Hola";` ó `$var = 'Hola';`
- **Boolean:** los que toman dos valores True (1) ó False (0)
`$var = TRUE` ó `$var = FALSE;`
- **Matrices:** son los arrays, permiten almacenar valores múltiples bajo un mismo nombre de datos y que pueden tener distintos tipos de datos.
`$matriz[0]="La Coruña";`
`$matriz[1]="Las Palmas";`
`$matriz[2]="Tenerife";`

No es obligado que todos los elementos del array sean del mismo tipo

```
$matriz[3] = 100;
```

Se pueden definir valores aunque no se hayan definido los anteriores, así se puede definir

```
$matriz[5] = "Hola" sin haber definido $matriz[4].
```

Para insertar en la última posición se usa `$matriz[] = "Al final"`.

Las matrices pueden ser asociativas donde los índices en vez de ser números son cadenas de caracteres.

```
$matrizAsoc["España"] = "Madrid";  
$matrizAsoc["Italia"] = "Roma";  
$matrizAsoc["Francia"] = "París";
```

- **NULL:** es el tipo de una variable sin asignación

Representa una variable sin valor asignado. Una variable puede ser nula por tres razones:

- Se le asignó un valor NULL.
- Todavía no se le ha asignado un valor
- Se anuló con la función unset()

Para asignar \$var = NULL;

Veamos el siguiente ejemplo

```
$Cadena = "Tipo de dato de cadena";
$NúmeroEntero = 1; // Un valor entero
$NúmeroFlotante = 1.55; // Un valor numérico con decimales
$Booleano = True; // Un valor booleano True (1) o False (0)
$Matriz[0] = "A"; // Un valor de matriz con subíndice 0
$Matriz[2] = 3; // Un valor de matriz con subíndice 2

$NúmeroOctal = 012; // octal 12 es decimal 10
$NúmeroHexadecimal = 0x1C; // hexadecimal 1c igual a decimal 28
$NúmeroNegativo = -33; // números negativos llevan el signo adel
$NúmeroFlotanteExp = 1.55e3;

// impresión de los datos
echo "Cadena:" . $Cadena . "<BR>";
echo "Número entero:" . $NúmeroEntero . "<BR>";
echo "Número flotante:" . $NúmeroFlotante . "<BR>";
echo "Booleano:" . $Booleano . "<BR>";
echo "\$Matriz[0]:" . $Matriz[0] . "<BR>";
echo "\$Matriz[2]:" . $Matriz[2] . "<BR>";
echo "Número octal:" . $NúmeroOctal . "<BR>";
echo "Número hexadecimal:" . $NúmeroHexadecimal . "<BR>";
echo "Número negativo:" . $NúmeroNegativo . "<BR>";
echo "Exponencial:" . $NúmeroFlotanteExp . "<BR>";
```

Fichero: 08-tipos_de_datos.php

Sentencias condicionales en PHP

Los condicionales se usan para comprobar una condición determinada y según sea el resultado (VERDAD ó FALSO) de esa evaluación se decide hacer una acción u otra.

PHP suministra dos sentencias condicionales: if y switch

INSTRUCCIÓN IF

El formato es el siguiente

```
if (condicion_a_evaluar){  
    acciones a ejecutar si se cumple la condición (es verdad o true)  
}  
else {  
    acciones a ejecutar si no se cumple la condición (es falsa)  
}
```

La opción del else es opcional, es decir, no es obligatorio que aparezca. Si sólo se va a ejecutar una instrucción las { } se pueden quitar, pero si se va a ejecutar más de una son de uso obligatorio para agruparlas.

Cuando la instrucción que se va a ejecutar en el apartado del “else” es otro “if”, existe una forma abreviada para expresarlo “elseif” y la instrucción quedaría como sigue

```
if (condicion_a_evaluar){  
    acciones a ejecutar si se cumple la condición (es verdad o true)  
}  
elseif (otra_condicion_a_evaluar) {  
    acciones a ejecutar si se cumple la otra condición  
}  
else {  
    acciones a ejecutar si no se cumple la otra condición  
}
```

Las condiciones que se evalúan pueden tener los distintos operadores de comparación siguientes

\$oper1 comparador \$oper2

donde el comparador puede ser

> para mayor que

< para menor que
>= para mayor o igual que
<= para menor o igual que
== para igual que (el mismo valor aunque no el mismo tipo)
=== para igual que (el mismo valor y el mismo tipo de datos)
<> ó != para distinto que
!== inverso a ===

A veces hay que preguntar por mas de una condición dentro del condicional, para ello se concatenan las distintas opciones mediante los operadores lógicos

\$oper1 operadorlógico \$oper2 ó
operadornegacion \$oper1

Los operadores lógicos son

A AND B ó A && B (Y lógico)
es verdad cuando A Y B SON VERDAD Y FALSO CUANDO UNA
DE LAS DOS LO ES

A OR B ó A || B (O lógico)
es verdad cuando A ó B SON VERDAD Y FALSO CUANDO LAS
DOS SEAN FALSAS

A XOR B (XOR lógico)
es verdad cuando A ó B SON VERDAD pero no al mismo tiempo
y es falsa cuando las dos son falsas o las dos verdaderas

NOT A ó !A (Negación Lógica)
es verdad cuando A es FALSO

Ver ejemplo 09-if.php

LA INSTRUCCIÓN SWITCH

El operador switch, es un condicional que permite comparar el valor de una variable con varios valores preestablecidos, permitiendo ejecutar distintas acciones dependiendo del valor al que se haya igualado. Si el valor de la variable de comparación no es igual a ninguno de esos valores, se ejecutará las opciones por defecto si estas fueron establecidas en la instrucción.

el formato del operador switch es el que se muestra a continuación

```

switch ( $variable ) {
    case valor 1:
        instrucciones a ejecutar para valor 1
        break;
    case valor 2:
        instrucciones a ejecutar para valor 2
        break;
    .
    .
    .
    case valor n:
        instrucciones a ejecutar para valor n
        break;
    default:
        instrucciones a ejecutar para el default
}

```

Es posible ejecutar el mismo bloque de acciones para varios de los valores establecidos, en ese caso se pueden agrupar las opciones.

El tipo de datos de la variable a comparar puede ser un entero o un string, pero no puede tomar valores decimales (float)

Ver como ejemplo el fichero 10-switch.php

BUCLES EN PHP

Una situación habitual en la lógica de los programas es tener que ejecutar repetidamente un conjunto de instrucciones hasta que se deje de cumplir una determinada condición. A este tipo de instrucciones es lo que se conoce con el nombre de bucles.

Los bucles que existen en PHP son los siguientes

- while
- for
- do...while
- foreach (sólo para los array)

EL OPERADOR WHILE EN PHP

El while en PHP tiene el siguiente formato

```

while (condicion_a_evaluar) {

    sentencias a ejecutar mientras se cumpla la condición ( sea verdad )

    +

    sentencia que modifique la variable involucrada en la condición

}

```

Lo que quiere decir que las instrucciones dentro del while se ejecutarán siempre que se cumpla la condición del mismo.

SIEMPRE SE PREGUNTA PRIMERO ANTES DE EJECUTAR Y SI NO SE CUMPLE LA CONDICIÓN NO SE EJECUTA NADA

En los bucles pueden existir ocasiones, en las que queremos que se cumplan varias condiciones para poder ejecutar un conjunto de instrucciones. En esos casos esas condiciones deberán estar unidas por operadores AND ó OR ó NOT.

Podemos tener dos maneras de salir de un bucle bien por el control de una variable booleana o mediante la sentencia break.

ejemplo usando una variable booleana

```

$encontre = FALSE;

$i = 1; $n = 100

while ((! $encontre) AND ( $i < $n )) {
    if ( condicion ) {
        ...
        $encontre = TRUE; Hemos encontrado lo que buscamos y queremos salir
    }
    $i = $i + 1; incrementamos el contador para trabajar la próxima iteración
}

```

aquí hay que preguntar de que manera (por cual condición has salido del bucle)

```

if ( $i == $n ) {      no encontro nada, hubiera sido igual preguntar ( ! $encontre )
    acciones a realizar si no encuentre lo buscado
}

```

ejemplo usando la instrucción brake

```
$i = 1; $n = 100
while ($i < $n) {
    if ( condicion ) {
        ...
        break; Hemos encontrado lo que buscamos y queremos salir
    }
    $i = $i + 1; incrementamos el contador para trabajar la próxima iteracion
}
if ( $i == $n ) { no encontro nada

    acciones a realizar si no encuentre lo buscado

}
```

la instrucción break nos permite salir de los bucles anidados

```
while ( cond1 ) {
    ...
    while ( cond2 ) {
        ...
        break(2); break(2) sale de los dos bucles y se ejecutan las
                    instrucciones marcadas con (2)
        break(1) sale del bucle interior y se ejecutan las
                    instrucciones marcadas con (1)
    }
    (1)...
}
(2) ...
```

EL OPERADOR FOR EN PHP

El segundo tipo de bucle que vamos a estudiar es el bucle for.

El bucle for en PHP tiene el siguiente formato

```
for ( valor inicial contador; condicion a cumplir; modificación contador) {

    acciones a realizar en cada iteración del bucle

}
```

```
}
```

Cuando se ejecuta el for por primera vez se toma el valor inicial del contador esto sólo se realiza la primera vez. Luego se pregunta si se cumple la condición si la condición se cumple se ejecutan las acciones a realizar, por último se realiza la operación de modificación del contador, antes de volver a preguntar si se cumple la condición. Las acciones del for se ejecutan siempre y cuando se cumpla la condición.

El problema de los for es que las tres partes que lo componen NO son obligatorias, pudiendo tener un for de esta manera

```
for ( ; ; ) {  
  
    acciones a ejecutar  
  
}
```

en este caso en las acciones a ejecutar tendrá que usarse un break para salir del bucle infinito.

Ejemplo de escribir los números del 1 al 10 con while

```
echo "<br />números escritos mediante while<br /><br />";  
$i = 1;  
while ( $i <= 10) {  
    echo "$i ";  
    $i = $i + 1;  
}
```

Ejemplo de escribir los números del 1 al 10 con for

```
echo "<br /><br />números escritos mediante for<br /><br />";  
for ($i = 1; $i <= 10; $i = $i + 1) {  
    echo "$i ";  
}
```

En los bucles for pueden existir varias condiciones por las que queramos salir de el. En esos casos esas condiciones estarán unidas por operadores AND ó OR. Podemos tener dos maneras de salir de un bucle bien por el control de una variable booleana o mediante la sentencia break.

ejemplo de for con varias condiciones usando una variable booleana

tener cuidado con los errores lógicos

```
$n = 100;
for ($encontre = FALSE, $i = 1; (! $encontre) AND ($i < $n); $i = $i + 1) {
    if ( $i > 50 ) {
        echo "<br />Encontre el valor buscado $i <br />";
        $encontre = TRUE; Hemos encontrado lo que buscamos y queremos salir
    }
}
if ( $i == $n ) { no encontro nada, hubiera sido igual preguntar ( ! $encontre )
    echo "<br />No encontré nada<br />";
}
```

ejemplo usando la instrucción brake

```
$n = 100
for ( $i = 1; $i < $n; $i = $i+1) {
    if ( condicion ) {
        ...
        break; Hemos encontrado lo que buscamos y queremos salir
    }
    $i = $i + 1; incrementamos el contador para trabajar la próxima iteracion
}
if ( $i == $n ) { no encontro nada

    acciones a realizar si no encuentre lo buscado

}
```

la instrucción break nos permite salir de los bucles anidados

```
for ( $i = 1; $i < $n; $i++ ) {
    ...
    for ( $j = 1; $j <= $n; $j++ ) {
        ...
        break(2); break(2) sale de los dos bucles y se ejecutan las
                    instrucciones marcadas con (2)
        break(1) sale del bucle interior y se ejecutan las
                    instrucciones marcadas con (1)
    }
    (1)...
}
(2) ...
```

EL OPERADOR DO..WHILE

El do while en PHP tiene el siguiente formato

```
do {  
    sentencias a ejecutar mientras se cumpla la condición ( sea verdad )  
    sentencia que modifique la variable involucrada en la condición  
    en el do while estas instrucciones se ejecutan al menos una vez  
  
} while (condicion);
```

Lo que quiere decir que las instrucciones dentro del do while se ejecutarán siempre que se cumpla la condición del mismo.

SIEMPRE SE EJECUTA UNA VEZ ANTES DE PREGUNTAR LA CONDICION, SI NO SE CUMPLE YA NO SE EJECUTA MAS PORQUE SE SALE DEL CICLO

Ejemplo

Tenemos un array de 6 elementos y queremos imprimir sus valores en pantalla

```
$x[0] = 1;  
$x[1] = 2;  
$x[2] = 3;  
$x[3] = 4;  
$x[4] = 5;  
$x[5] = 6;
```

sin DO while

```
echo "Los valores que están en el array son: ";  
echo "$x[0] ";  
echo "$x[1] ";  
echo "$x[2] ";  
echo "$x[3] ";  
echo "$x[4] ";  
echo "$x[5]<br /><br />";
```

usando un DO while

```
echo "Los valores que están en el array usando do while: ";  
$i = 0;  
do {
```



```
        echo "$x[$i] ";  
        $i = $i + 1;  
    } while ( $i <= 5)
```

vale lo mismo que se explico para el while en lo referente a las condiciones con varias opciones y los break

`/* TIPOS DE DATOS EN PHP`

En PHP existen los siguientes tipos de datos:

integer -- los números enteros

float o double -- los números decimales

boolean -- que solo tiene dos valores TRUE (<>0) ó FALSE(=0)

NULL -- es el tipo de una variable sin asignación

arrays o matrices -- Permiten almacenar varios valores bajo una misma denominación

string -- las cadenas de caracteres de ilimitada longitud

object -- objetos tipos de datos complejos que se crean a partir de una clase

resources ó recursos -- son variables especiales que mantienen referencias a recursos externos por ejemplo una conexión a BD

En este fichero hablaremos de los 4 primeros tipos

NÚMEROS ENTEROS (integer)

Los números enteros se pueden poner en decimal, octal o hexadecimal

Los números enteros son números sin decimales que pueden ser positivos o negativos. Se pueden expresar en base decimal, octal o hexadecimal

```
$valor = 20; // decimal
```

```
$valor = 024; // octal
```

```
$valor = 0x14; // hexadecimal
```

```
$valor = 1100101;
```

```
$valor = -24;
```

Los enteros ocupan 4 bytes, el primer bit es para el signo y su rango va desde el -2.147.483.648 al +2.147.483.647.

Si en una variable entera se asigna un valor mayor al que puede soportar se convierte automáticamente en float

NÚMEROS DECIMALES (float, double, real)

Son los que permiten almacenar números con decimales. Ocupan 8 bytes y su rango es del $2.2E-308$ al $1.8E+308$.

La parte entera se separa de la decimal mediante el '.' y también se puede expresar en formato exponencial.

```
$valor = 250.89;  
$valor = -25089e-2; // que corresponde al -250,89
```

TIPO BOOLEANO

El tipo booleano sólo tiene dos valores TRUE y FALSE.

Hay que tener cuidado con los valores booleanos porque con la conversión de tipos implícita que realiza PHP, muchos de los condicionales terminan siendo comparaciones de booleanos.

veamos este ejemplo

```
$n = 1;  
$d = 5;  
if ($d == 0) {  
    print "Se ejecutó la instrucción dentro del if\n";  
}
```

pero también podemos tener este código

```
if ( $d ) {  
    echo "nada"  
}  
else {  
    print "Ahora se ejecuta la instrucción del else\n";  
}
```

en este último ejemplo no hay operador de comparación y la conversión de tipos hace que se escriba "nada" ya que el 5 es $\neq 0$ y se considera TRUE.

Los valores que PHP considera TRUE para los distintos tipos es:

tipo de datos	Es TRUE cuando	ES FALSE cuando
-----	-----	-----
entero	<> 0	= 0
float	<> 0.0	= 0.0
string	al menos un caracter	"" ó "0"
array	al menos un elemento	vacio
NULL	nunca	siempre
boolean	TRUE, true	FALSE, false
object	siempre	nunca
resources	siempre	nunca

TIPO NULL

El tipo NULL es un tipo de dato con único valor el NULL. Este valor indica que la variable está vacía (que no tiene valor). Es muy útil para diferenciar entre el string vacío "" y el valor null de un campo de base de datos.

\$valor = NULL;

OPERACIONES CON LOS TIPOS DE DATOS NUMÉRICOS

+ suma
- resta
* multiplicación
/ división (con decimales)
% resto de la división entera
= operador asignación --- \$valor = 0;

+= asignar suma
-= asignar con resta
*= asignar con multiplicación
/= asignar con división
%= asignar con módulo

\$x += 1 es lo mismo que hacer \$x = \$x + 1;
\$x -= 1 es lo mismo que hacer \$x = \$x - 1;
\$x *= 2 es lo mismo que hacer \$x = \$x * 2;
\$x /= 2 es lo mismo que hacer \$x = \$x / 2;
\$x %= 2 es lo mismo que hacer \$x = \$x % 2;

operador ++

`$x++` es lo mismo que hacer `$x = x + 1`

`++$x` es lo mismo que hacer `$x = x + 1`

La única diferencia entre las dos es que en la primera opción primero se usa el valor y luego se incrementa en 1, en la segunda primero se incrementa en 1 y luego se usa el valor

```
$x = array(1,2,3);  
$i = 0;  
$valor = $x[$i++];  
echo "$valor<br />";
```

operador --

`$x--` es lo mismo que hacer `$x = x - 1`

`--$x` es lo mismo que hacer `$x = x - 1`

La única diferencia entre las dos es que en la primera opción primero se usa el valor y luego se decrementa en 1, en la segunda primero se decrementa en 1 y luego se usa el valor

```
$x = array(1,2,3);  
$i = 0;  
$valor = $x[--$i];  
echo "$valor<br />";
```

```
$i = 3;  
while ( ($i > 0) and ($x[--$i] < 0))
```

EL OPERADOR CAMBIO DE TIPO CAST (tipodato)

El operador CAST permite cambiar de tipo entre aquellos tipos que pueden ser compatibles. El modo de usarlo es el siguiente

```
$variable = (tipodato)(expresion);
```

donde tipodato puede ser integer, float, string, bool, array, object

ejemplo

`$valor = 7/3;` en este caso `$valor = 2.6666666` un valor decimal

`$valor = (int)(7/3)` en este caso como se fuerza a entero `$valor = 2`

```
$palabra = "45";  
$numero = (int)$palabra;
```

ARRAYS EN PHP

el array es la manera que hay para tener gran cantidad de datos bajo una misma denominación en la que sólo se difiere en la posición (índice)

un ejemplo de array de números enteros

```
$x[0] = 1;  
$x[1] = 2;  
$x[2] = 3;  
$x[3] = 4;  
$x[4] = 5;  
$x[5] = 6;
```

un ejemplo de array de string

```
$s[0] = "hola";  
$s[1] = "como";  
$s[2] = "estas";
```

en los dos casos anteriores el índice es un número entero. En los array el índice siempre va del valor 0 hasta el número de elementos - 1, es decir que en los ejemplos anteriores

el primer elemento es `$x[0]` y `$s[0]`
y el último elemento es `$x[5]` y `$s[3]`

otra forma de llenar el array s anterior puede ser

```
$s1[] = "hola";  
$s1[] = "como";  
$s1[] = "estas";
```

obteniendo el mismo resultado que antes

hay que tener cuidado porque los arrays en PHP no tienen que tener elementos consecutivos

```
$x1[0] = 10;  
$x1[20] = 134;  
$x1[5] = 20;  
$x1[10] = 30;
```

en este caso el array tiene 3 elementos en las posiciones 0, 5 y 10

a la hora de escribir los elementos del último array no se podrá hacer de la manera convencional ya que dará error

```
echo "<br />ejemplo de escritura array con elementos no contiguos<br /><br />";  
$i=0;  
$nroelementos = count($x1);  
while ( $i < $nroelementos ) {  
    print "posición $i tiene el valor $x1[$i]";  
    $i++;  
}  
echo "<br />";
```

en estos casos se debe utilizar el bucle foreach
la forma de utilizar el foreach es la siguiente

```
foreach (array as $nombre_variable) {  
  
    las instrucciones a ejecutar UTILIZAN $nombre_variable  
  
}
```

```
echo "<br />usando el foreach con un array con índice numérico<br />";  
  
foreach ($x1 as $aux) {  
    print "este elemento tiene el valor $aux<br />";  
}
```

los elementos de un array no tienen porque ser del mismo tipo
es decir sus elementos pueden ser de diferente tipo.

```
$x2[0] = 1;  
$x2[1] = "abc";  
$x2[2] = 17.50;
```

```
echo "<br />usando el foreach con array de <> elementos<br />";  
$nroelemento = 1;  
foreach ($x2 as $aux) {  
    print "el elemento tiene el valor $aux<br />";  
    $nroelemento++;  
}
```

en los array los índices no tienen porque ser numéricos
sino que también pueden ser string -- arrays asociativos

```
$links = array("The Apache Web Server" => "www.apache.org",  
               "Apress" => "www.apress.com",  
               "The PHP Scripting Language" => "www.php.net");
```

```
echo "<br />Mostrando un array asociativo<br />";  
echo "<br />Direcciones web:<br /><br />";  
foreach($links as $title => $aux) {  
    echo "$title *** $aux<br />";  
}
```

ejemplo

```
$Nombre[0]="Jose";  
$Nombre[10]="Alberto";  
$Nombre[20]="Raul";  
$Nombre[30]="Pedro";  
$Nombre[40]="Emilio";  
$Nombre[50]= 13; un array no es de un tipo determinado  
$Nombre[] = "Al final posicion 6"; agrega un elemento al final  
$Nombre[80]="El ultimo en pos 80";
```

```
/*  
$i = 0;  
while ($i <= 7)  
{  
    print "El nombre en posicion i es = $Nombre[$i]<br />";  
}
```



```

$i++;
}
*/
$i=0;
foreach ($Nombre as $aux)
{
    print "El en la posicion $i es = $aux<br/>";
    $i++;
}

```

funciones de php para array

FUNCION COUNT()

la funcion count devuelve el nro de elementos que tiene el array

```

echo "<br />Probando la función count()<br />";
$x2[0] = 1;
$x2[1] = "abc";
$x2[2] = 17.50;
$nroelemento = count($x2);
echo "<br />El número de elementos del array x2 es = $nroelemento<br />";

```

FUNCION SORT()

la función sort() ordena de manera creciente los elementos de un array

```

echo "<br />Probando la función sort()<br />";
$x2[0] = 15;
$x2[1] = 3;
$x2[2] = 17;
echo "<br />El array antes de ordenar<br />";

foreach ($x2 as $aux) {
    print "$aux ";
    $nroelemento++;
}
echo "<br />";
sort($x2);
echo "<BR />El array después de ordenar<br />";

```

```
foreach ($x2 as $aux) {
    print "$aux<br />";
    $nroelemento++;
}
```

FUNCION ARRAY()

la función array() permite crear un array con índice numérico o con índice asociativo. la forma de utilizar es:

```
$x2 = array(13,56,23); crea un array de tres elementos con índice numérico
$sexo = array("hombre", "mujer"); crea un array de dos elementos
$idiomas = array( "España" => "español",
                  "Estados Unidos"=> "inglés",
                  "Francia" => "francés",
                  "China" => "chino");
```

FUNCTION RANGE()

la función range permite generar una secuencia de números desde un valor inicial hasta un valor final, incrementándolo en un valor determinado

la función range tiene el siguiente formato

```
range(valorinicial, valorfinal [,incremento])
```

```
$x2 = range(3,15,2); (3,5,7,9,11,13,15)
```

también se puede utilizar para generar una secuencia de caracteres

```
$aux = range('A','F'); (A,B,C,D,E,F)
```

FUNCIÓN PRINT_R()

esta función permite escribir un array en su totalidad, aunque sólo se debe utilizar para testeo ya que el formato de salida no es bueno

```
$aux = range('A','D'); (A,B,C,D)
echo "<br />uso de print_r<br />";
print_r($aux);
```

```
echo "<br />";
```

FUNCIÓN IS_ARRAY()

la función `is_array()` sirve para preguntar si una variable es de tipo array. En caso afirmativo devuelve TRUE y falso en el caso contrario

```
echo "<br />probando la función is_array<br />";  
$states = array("Florida", "New York");  
$state = "Ohio";  
printf("\$states is an array: %s <br />",  
(is_array($states)) ? "TRUE" : "FALSE";  
printf("\$state is an array: %s <br />",  
(is_array($state)) ? "TRUE" : "FALSE");
```

Aquí hemos introducido un nuevo operador

el operador `?` : funciona de la siguiente forma

condicion_a_evaluar ? instruccion si verdad : instruccion si falso

FUNCIÓN ARRAY_UNSHIFT()

la función `array_unshift()` permite añadir un elemento al comienzo de un array. Si los índices son numéricos se añade 1 a cada uno de ellos, siendo el nuevo elemento el que tiene el índice de valor 0.

el prototipo de la función es el que se muestra a continuación

```
int array_unshift(array, valor_añadir [, valor_añadir...])
```

```
echo "<br />probando array_unshift<br />";  
$states = array("Ohio","New York");  
array_unshift($states,"California","Texas");  
  
foreach ($states as $aux) {  
    print "$aux<br />";  
}
```

FUNCIÓN ARRAY_PUSH()

la función `array_push()` añade un elemento al final del array

el prototipo de la función es el que se muestra a continuación

`int array_push(variable_array, valor_añadir [, valor_añadir...])`

```
echo "<br />Probando array_push<br />";
$states = array("Ohio","New York");
array_push($states,"California","Texas");
foreach ($states as $aux) {
    print "$aux<br />";
}
```

FUNCIÓN `ARRAY_SHIFT()`

la función `array_shift` elimina y retorna el primer elemento del array, reduciendo en 1 el índice del resto de elementos que se quedan en el array

el prototipo de la función es el que se muestra

`$variable = array_shift(array)`

```
echo "<br />probando array_shift<br />";
$states = array("Ohio","New York","California","Texas");
$state = array_shift($states);
foreach ($states as $aux) {
    print "$aux<br />";
}
echo "<br />el valor de la variable \$state = $state<br />";
```

FUNCIÓN `ARRAY_POP()`

la función `array_pop` elimina y retorna el último elemento del array

el prototipo de la función es el que se muestra

`$variable = array_pop(array)`

```
echo "<br />probando array_pop<br />";
$states = array("Ohio","New York","California","Texas");
$state = array_pop($states);
```

```
foreach ($states as $aux) {
    print "$aux<br />";
}
echo "<br />el valor de la variable \$state = $state<br />";
```

FUNCIÓN IN_ARRAY()

la función in_array() busca un elemento dentro de un array devuelve TRUE si lo consigue y FALSE sino

in_array(valor_buscar, array [, verifica_tipo])

verifica tipo un valor booleano si quiere que se compruebe tipo

```
echo "<br />probando in_array<br />";
$state = "Ohio";
$states = array("California", "Hawaii", "Ohio", "New York");
if (in_array($state, $states)) {
    echo "el valor buscado estaba en el array<br />";
}
```

FUNCIÓN ARRAY_KEY_EXISTS()

la función array_key_exists es igual a la anterior pero con array asociativos

su prototipo es el siguiente

boolean array_key_exists(clave_a_buscar, array)

```
echo "<br />probando array_key_exists<br />";
$estado["Delaware"] = "December 7, 1787";
$estado["Pennsylvania"] = "December 12, 1787";
$estado["Ohio"] = "March 1, 1803";
if (array_key_exists("Ohio", $estado)) {
    printf("Ohio joined the Union on %s<br />", $estado["Ohio"]);
}
```

FUNCIÓN ARRAY_SEARCH()

la función array_search() busca un elemento en un array asociativo, si lo encuentra retorna la clave donde está ese elemento en el array y sino retorna FALSE

el prototipo es el siguiente

```
$valor_devuelto = array_search(valor_buscar, array [, strict])
```

el parámetro strict es booleano y si se pone a TRUE es que queremos también la igualdad de tipos

```
echo "<br />probando array_search<br />";
$estado2["Ohio"] = "March 1";
$estado2["Delaware"] = "December 7";
$estado2["Pennsylvania"] = "December 12";
$founded = array_search("December 7", $estado2);
if ($founded)
    printf("%s was founded on %s.<br />", $founded, $estado2[$founded]);
```

ARRAYS EN DOS DIMENSIONES

Ejemplos de array de dos dimensiones los podemos tener en el plano cartesiano, en el tablero de ajedrez, en la estadísticas de dos dimensiones, almacenar los datos de fotos, etc.

por eso es útil los arrays de dos dimensiones (matrices)

Una matriz como la siguiente

(5) columnas el índice va de 0..4

	1	3	5	8	2
3 filas	3	9	0	3	5
0..2	8	1	0	7	6

donde el elemento 7 de la tercera fila y cuarta columna se puede se puede referenciar como matriz[2][3]

como se puede apreciar ahora se tiene otro par de [] para la nueva dimensión.

```
$matriz[ fila ][ columna ]
```

la matriz anterior se llena de forma manual de la siguiente forma

```
$matriz[0][0] = 1;
$matriz[0][1] = 3;
$matriz[0][2] = 5;
$matriz[0][3] = 8;
$matriz[0][4] = 2;
$matriz[1][0] = 3;
$matriz[1][1] = 9;
$matriz[1][2] = 0;
$matriz[1][3] = 3;
$matriz[1][4] = 5;
$matriz[2][0] = 8;
$matriz[2][1] = 1;
$matriz[2][2] = 0;
$matriz[2][3] = 7;
$matriz[2][4] = 6;
```

el procedimiento para crear la matriz anterior usando la instrucción array

```
$matriz = array( array( 1, 3, 5, 8, 2 ), array( 3, 9, 0, 3, 5 ), array( 8, 1, 0, 7, 6 ) );
```

Es obvio que para recorrer una matriz como la anterior se debe usar dos bucles anidados

Si queremos usar los bucles condicionales while, for, do while deberemos averiguar primero el número de filas y de columnas que tiene la matriz
Para ello debemos usar la función count()

Para averiguar el número de filas se pasa a la función count()
la variable que representa la matriz

Para averiguar el número de columnas se pasa a la función la fila a la que se quiera contar los elementos

```
echo "<br />Mostrando el nro de filas y columnas de la matriz<br />";
$nfil = count($matriz);
$ncol = count($matriz[0]);
echo "la matriz \$matriz tiene $nfil filas y $ncol columnas<br />";
```

El recorrido de la matriz para escribirla es el que se muestra a continuación

```
echo "<br />Los elementos de la matriz son <br /><br />";
$i = 0; se moverá por las filas
```



```

while ( $i < $nfil ) {
    $j = 0;
    $ncol = count($matriz[$i]);
    while ( $j < $ncol ) {
        echo $matriz[$i][$j]." ";
        $j++;
    }
    echo "<br />";
    $i++;
}

```

Un ejemplo de una matriz multidimensional asociativa es el que se puede obtener si queremos representar los valores de esta tabla

First key	Second key	Value
Soda can	Shape	Cylinder
	Color	Red
	Material	Metal
Notepad	Shape	Rectangle
	Color	White
	Material	Paper
Apple	Shape	Sphere
	Color	Red
	Material	Fruit
Orange	Shape	Sphere
	Color	Orange
	Material	Fruit
Phonebook	Shape	Rectangle
	Color	Yellow
	Material	Paper

Para crearla

```

$objects=array(
    'Soda can' => array('Shape' => 'Cylinder',
        'Color' => 'Red',
        'Material' => 'Metal'),

    'Notepad' => array('Shape' => 'Rectangle',
        'Color' => 'White',
        'Material' => 'Paper'),

    'Apple' => array('Shape' => 'Sphere',

```

```

'Color' => 'Red',
'Material' => 'Fruit'),
'Orange' => array('Shape' => 'Sphere',
'Color' => 'Orange',
'Material' => 'Fruit'),
'Phonebook' => array('Shape' => 'Rectangle',
'Color' => 'Yellow',
'Material' => 'Paper'));

```

```

echo "<br /><br />Los elementos de la matriz asociativa son <br /><br />";
print_r($objects);

```

CONSTANTES EN PHP

Contrario a lo que veníamos trabajando con las variables, las constantes son zonas de memoria a las que también se le asigna un nombre, pero que su valor no cambia a durante la ejecución del script y no pueden ser redefinidas.

Es decir que su valor se mantiene constante.

Los nombres de las constantes NO empiezan con el signo \$ y aunque no es obligatorio se suelen poner nombres en mayúsculas.

Una vez que la constante se define su alcance es global (es decir que vale para todo el código a partir de donde se define).

La sintaxis para declarar una constante es:

```
define ("NOMBRE_DE_LA_CONSTANTE", valor, [SENSIBLE_MAY_MIN]);
```

donde

"NOMBRE_DE_LA_CONSTANTE" : es el nombre que se le dará a la constante

valor : es el valor que tomará esa constante

SENSIBLE_MAY_MIN : es un valor booleano OPCIONAL que indica si queremos que el nombre de la variable sea sensible a may/min (valor TRUE) o sólo como se ha definido (valor FALSE), por defecto el valor es TRUE.

```
define("PI", 3.1416, TRUE);
```

```
// recordamos que el área de un círculo es A = pi * r * r
```

```
$radio = 5;  
$area = PI * $radio * $radio;  
echo "el área del círculo es = $area<br /><br />";  
  
define("E", "€ (euros)" , FALSE);  
  
echo "El importe de la compra es 25".E."<br />";  
  
// pero la línea de abajo debe dar error;  
  
echo "El importe de la compra es 25".e."<br />";
```