
Introducción a la programación
de Aplicaciones Web en Entorno Servidor en PHP
XAMMP
HHP

INDICE

¿Qué te pedimos que hagas?

Esta tarea está compuesta de varios ejercicios a realizar en PHP cuyo objetivo es que desarrolles las destrezas básicas en este lenguaje. **Ten en cuenta que estas destrezas son fundamentales para el resto del módulo**, la tarea te resultará compleja al principio, pero es fundamental que intentes poner todo tu empeño en sacar adelante estos ejercicios de la forma más adecuada.

Todos los ejercicios se realizarán en una misma carpeta de proyecto (llamada `DWES01`). Es importante que leas atentamente las instrucciones de cada ejercicio y, dado que no se exige un entorno de desarrollo concreto, ni un sistema operativo concreto, es importante que:

- Todos los archivos tengan codificación UTF-8.
- Se usen rutas relativas a la hora de incluir o referenciar otro archivo. No deben usarse nunca rutas absolutas (dado que imposibilita la portabilidad de la aplicación web).

Si los archivos no tienen una codificación adecuada o si se usan rutas absolutas, la tarea será devuelta para su corrección.

Se valorará en todos los casos la corrección ortográfica y gramatical de los mensajes para comunicarnos con el usuario, así como la presentación clara de cualquier información que se muestre al usuario.

Realiza los siguientes ejercicios en un documento de texto llamado `dwes01/seccion1.doc` o `dwes01/seccion1.odt`.

1.1.- Sección 1

Orientaciones para el alumnado

En esta actividad se integran los siguientes criterios de evaluación:

- *RA1.a) Se han caracterizado y diferenciado los modelos de ejecución de código en el servidor y en el cliente Web.*
- *RA1.b) Se han reconocido las ventajas que proporciona la generación dinámica de páginas Web y sus diferencias con la inclusión de sentencias de guiones en el interior de las páginas Web.*
- *RA1.f) Se han verificado los mecanismos de integración de los lenguajes de marcas con los lenguajes de programación en entorno servidor.*
- *RA1.g) Se han reconocido y evaluado las herramientas de programación en entorno servidor.*

- *RA2.f) Se han utilizado directivas para modificar el comportamiento predeterminado.*
- *RA1.e) Se han identificado y caracterizado los principales lenguajes y tecnologías relacionados con la programación Web en entorno servidor.*
- *RA2.b) Se han identificado las principales tecnologías asociadas.*
- *RA1.c) Se han identificado los mecanismos de ejecución de código en los servidores Web.*
- *RA1.d) Se han reconocido las funcionalidades que aportan los servidores de aplicaciones y su integración con los servidores Web.*
- *RA2.a) Se han reconocido los mecanismos de generación de páginas Web a partir de lenguajes de marcas con código embebido.*

Realiza los siguientes ejercicios en un documento de texto llamado **`dwes01/seccion1.doc`** o **`dwes01/seccion1.odt`**.

Ejercicio 1 (RA1.a, RA1.b, RA1.d).

Seguro que visitas alguna web en las que siempre has planteado... "esto se puede mejorar" u "ojalá hicieran esto o aquello". Piensa en alguna de esas páginas y responde a las preguntas.

- ¿De qué negocio o empresa se trata? (indica la URL)

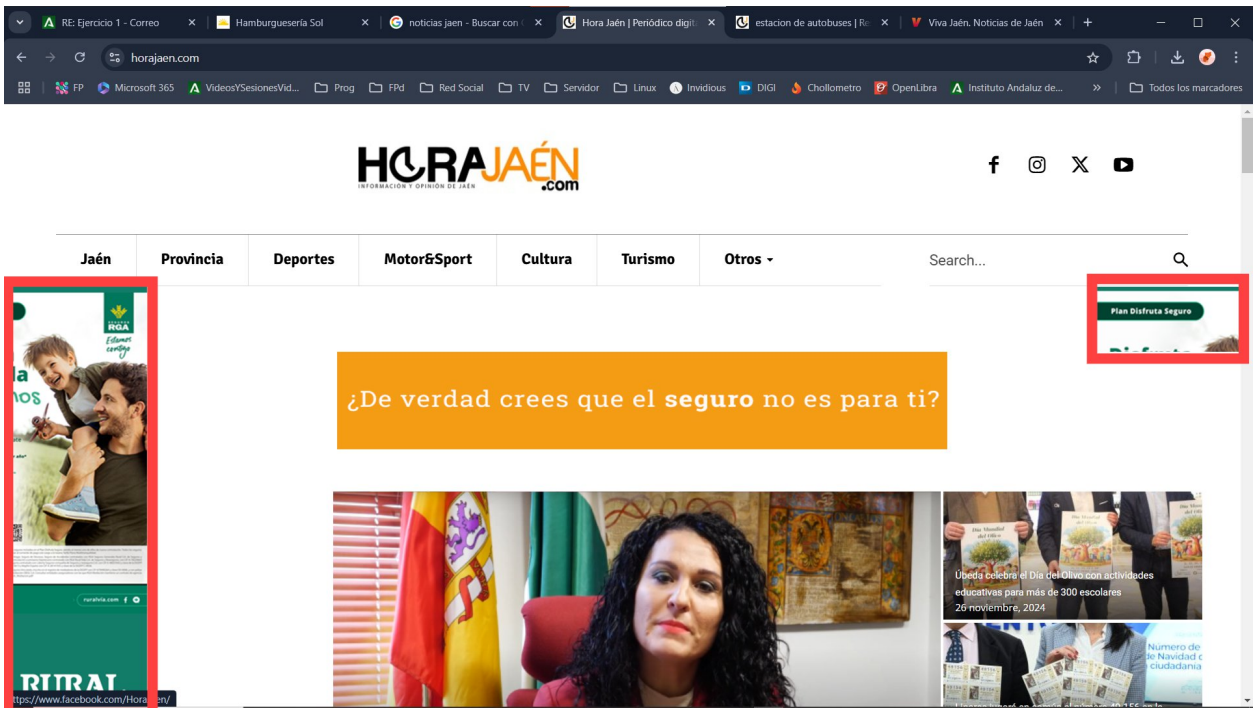
Se trata del diario digital horajaen, https://www.horajaen.com/.

- ¿Qué se puede hacer en su página web?

Consultar noticias que tienen que ver con la provincia de Jaén.

- ¿Qué aspectos negativos tiene su página web?

Contiene bastante publicidad, además no está bien encuadrada. He señalado dos zonas en las que puede apreciarse.



- ¿Tienen contenido dinámico (generación dinámica) o solo estático?

Tiene contenido dinámico, ya que por ejemplo genera de forma dinámica la búsqueda de noticias.

- Indica al menos dos mejoras que realizarías en dicha página que impliquen la ejecución de código en el servidor.

1. Filtrado de noticias descatacadas, por ejemplo de las noticias de la portada.
2. Un formulario de contacto

- Indica como podría ayudarte un Framework en la implementación de las mejoras anteriores.

Los frameworks PHP son piezas de software diseñadas para ayudar a los desarrolladores a crear aplicaciones de forma rápida y eficiente. Proporcionan una estructura predefinida para organizar el código, así como un conjunto de herramientas para agilizar el proceso de desarrollo, como puede ser la autenticación de usuarios, administración de sesiones y el almacenamiento caché.

Podemos encontrar framework que contenga alguna plantilla para la creación de un formulario web o bibliotecas para filtrar contenido (que serían las mejoras propuestas en el punto anterior).

Ejercicio 2 (RA1.a, RA1.c).

Fíjate en el siguiente ejemplo que describe la interacción en entre navegador (cliente web) y servidor web donde se detalla lo que ocurre en cada una de las partes. Intenta pensar en otro "Caso de uso" adaptado a la web elegida por ti en el ejercicio anterior donde un potencial usuario (o miembro de la organización) utilice la aplicación web en un contexto de generación dinámica de contenido.

Caso de uso: Un voluntario rellena el formulario de contacto

1. La persona voluntaria está visualizando un formulario en la página web de la asociación y ha terminado de rellenarlo, con lo que hace clic en "enviar".

2. El navegador realiza una petición tipo HTTP POST al servidor web.

3. El servidor web recibe la petición con los datos del formulario, y, siguiendo su configuración, detecta que esa petición corresponde a un script PHP y que debe ser redireccionada al motor de ejecución PHP.

4. El motor PHP (PHP engine) se arranca e inicia la ejecución del script el cual contiene:

- Código para comprobar si los datos recibidos son correctos.
- Código para almacenar los datos recibidos en la base de datos.
- Código para generar HTML con la respuesta al candidato a la candidata.

5. Una vez que el motor PHP ha terminado de ejecutar el script, el HTML generado es enviado al servidor web como resultado de la ejecución.

6. El servidor web recoge el resultado generado por el motor PHP y lo envía al navegador con una respuesta HTTP.

7. El navegador HTTP recibe la respuesta que incluye el HTML y lo renderiza para que la persona candidata lo vea.

1. El cliente está visitando la web para comprar un portátil hp i5 16Gb, por lo que se dirige al buscador de la página y escribe las palabras clave para facilitar la búsqueda, una vez escritas pulsa sobre "buscar".

2. El navegador realiza una petición tipo HTTP POST al servidor web.

3. El servidor web recibe la petición con los datos de la búsqueda, y, siguiendo su configuración, detecta que esa petición corresponde a un script PHP y que debe ser redireccionada al motor de ejecución PHP.

4. El motor PHP (PHP engine) se arranca e inicia la ejecución del script el cual contiene:

· Código para comprobar si los datos recibidos son correctos.

· Código para realizar una búsqueda en la base de datos.

· Código para generar HTML con la búsqueda realizada.

5. Una vez que el motor PHP ha terminado de ejecutar el script, el HTML generado es enviado al servidor web como resultado de la ejecución.

6. El servidor web recoge el resultado generado por el motor PHP y lo envía al navegador con una respuesta HTTP.

7. El navegador HTTP recibe la respuesta que incluye el HTML y lo renderiza para que el usuario de nuestra web pueda ver los resultados de su búsqueda.

Ejercicio 3 (RA1.e, RA1.g, RA2.b).

Rellena la siguiente tabla indicando **casos concretos** en los que podría ser necesario emplear las siguientes tecnologías en la mejora de la web elegida en el ejercicio 1:

Tecnología	¿Dónde y como se usaría?
HTTP	Se usa en el navegador web y lo usaríamos para comunicarnos con el servidor web e indicarle la página que queremos visualizar, la transmisión de datos de hace de forma no cifrada.
HTTPS	Se usa en el navegador web y lo usaríamos para comunicarnos con el servidor web e indicarle la página que queremos visualizar, la transmisión de datos de hace de forma cifrada.
HTML	Su uso principal es para la creación de páginas web, que posteriormente podemos visualizar a través de algún navegador web.
PHP	Es un lenguaje de programación que usamos del lado del servidor para crear sitios web (estáticos y dinámicos) o aplicaciones web. Un ejemplo de uso sería procesar un formulario web, si los datos son correctos, faltan datos o hay datos erróneos.
CSS	Se usa para dotar de estilo gráfico a las páginas web y sus distintos elementos, dotándolos de posición, forma, color, ... CSS se guarda en un archivo independiente y se referencia en la cabecera de las páginas HTML.
SQL	Se trata de un lenguaje completo de base de datos que usamos para realizar operaciones de selección, inserción, actualización,... Lo usaremos en nuestro servidor para consultar una o varias bases de datos, por ejemplo, al realizar una búsqueda de algún producto en una página web combinado con php.
JavaScript	Es un lenguaje de programación que usamos del lado del cliente para dotar de interactividad a las páginas web que visitamos. Un ejemplo de uso es la validación de campos dentro de un formulario, por ejemplo, el email o teléfono, sin necesidad de mandar el contenido al

	servidor y que sea este quien lo valide.

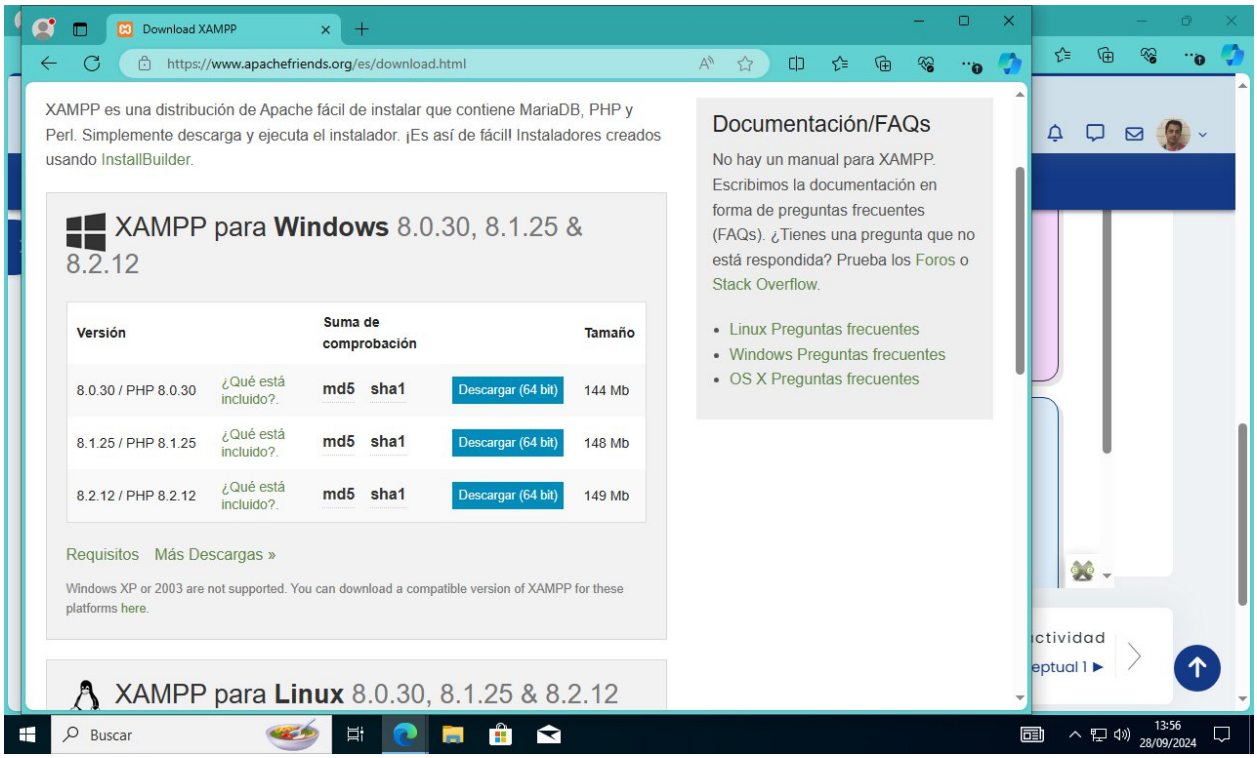
Ejercicio 4 (RA1.f, RA2.a, RA2.b).

Realiza la instalación de XAMPP 8.2.4 en tu ordenador (preferiblemente en `c:/xampp`) y realiza lo siguiente:

- Realiza una captura de pantalla durante el proceso de instalación.

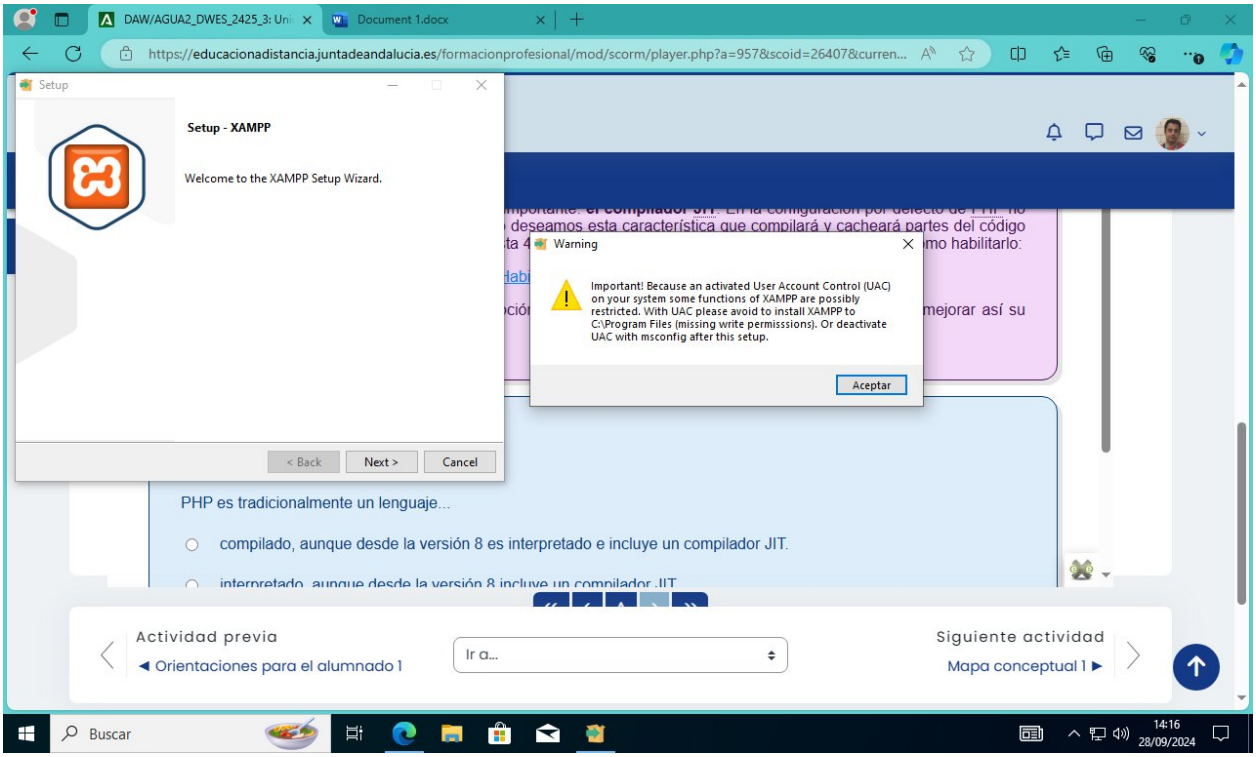
La instalación es bastante sencilla y parecida a la instalación de cualquier programa, lo primero que debemos hacer es escoger nuestra versión de XAMPP, en nuestro caso vamos a instalar la última versión, 8.2.12 para Windows de 64 bits, para ello pulsamos en el botón Descargar (64 bit), situado en la misma línea o pulsando el siguiente enlace

https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/8.2.12/xampp-windows-x64-8.2.12-0-VS16-installer.exe

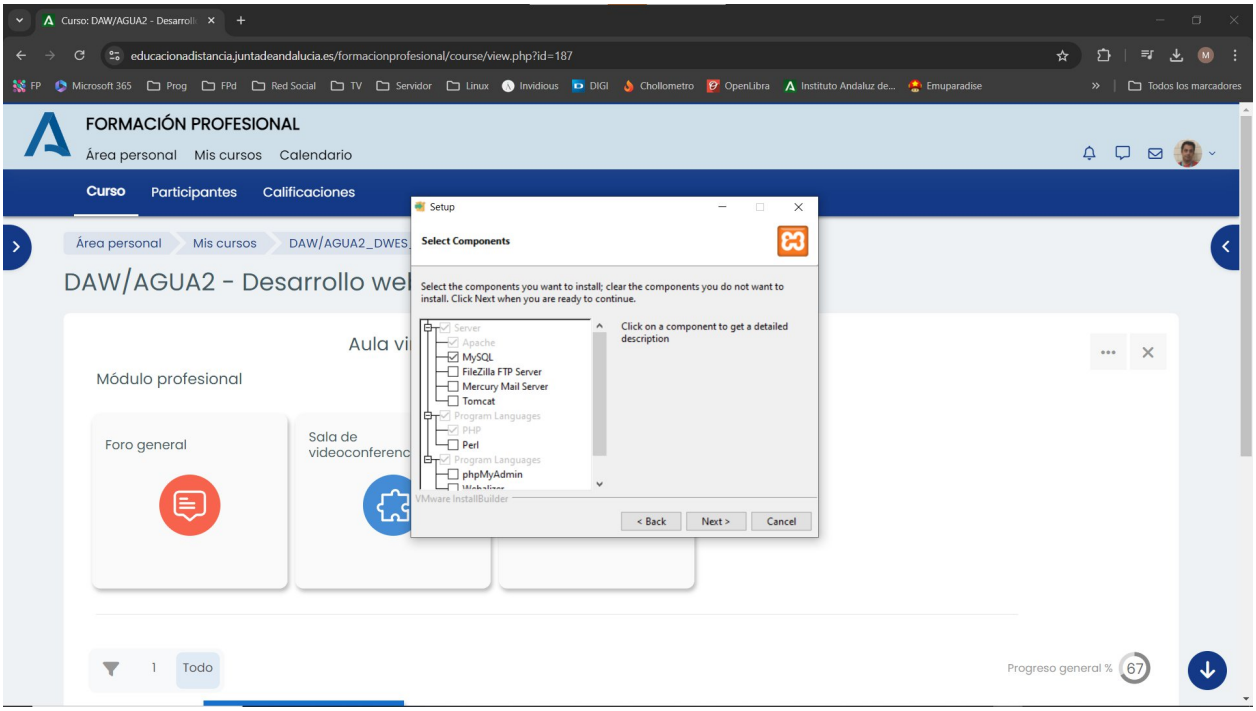


Una vez descargado, procedemos con la instalación.

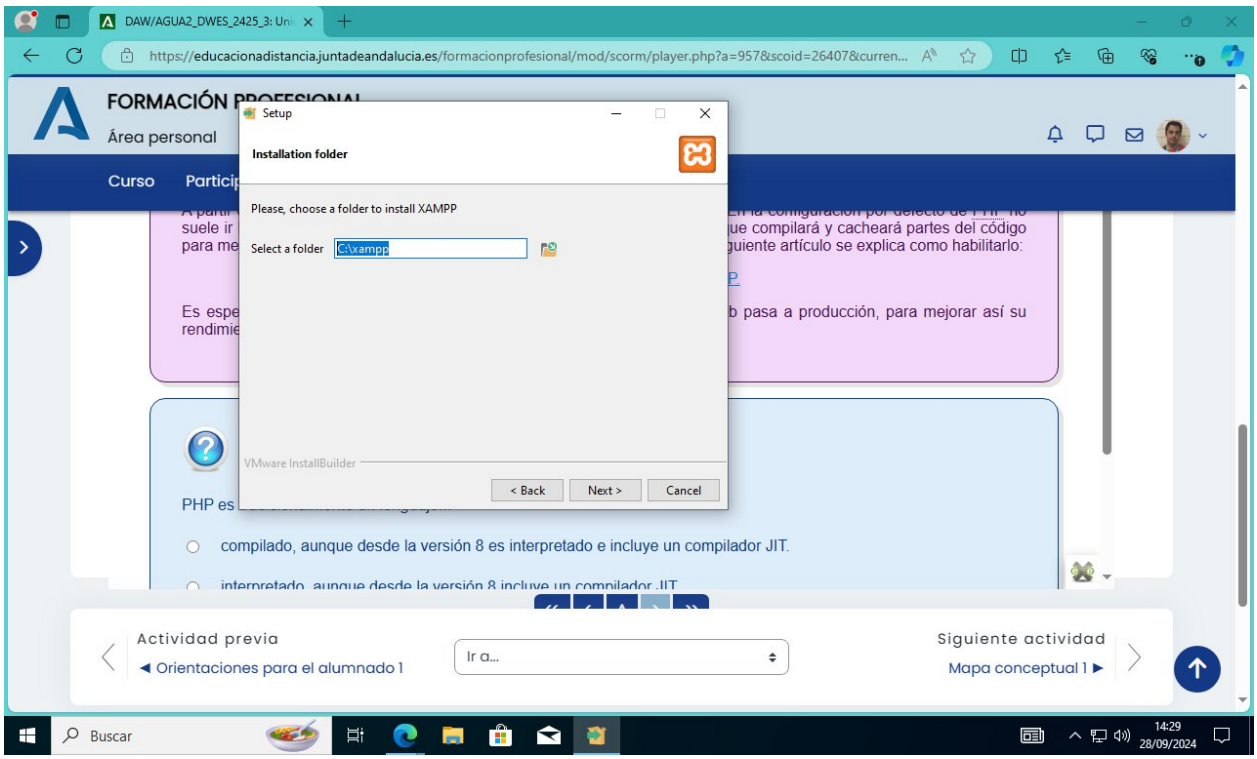
En la primera captura vemos que nos ha salido un aviso porque tenemos activado el control de cuentas UAC y nos pide que lo desactivemos al finalizar la instalación a través de msconfig (tecla Windows + R, en el cuadro de dialogo escribimos msconfig y pulsamos intro o aceptar).



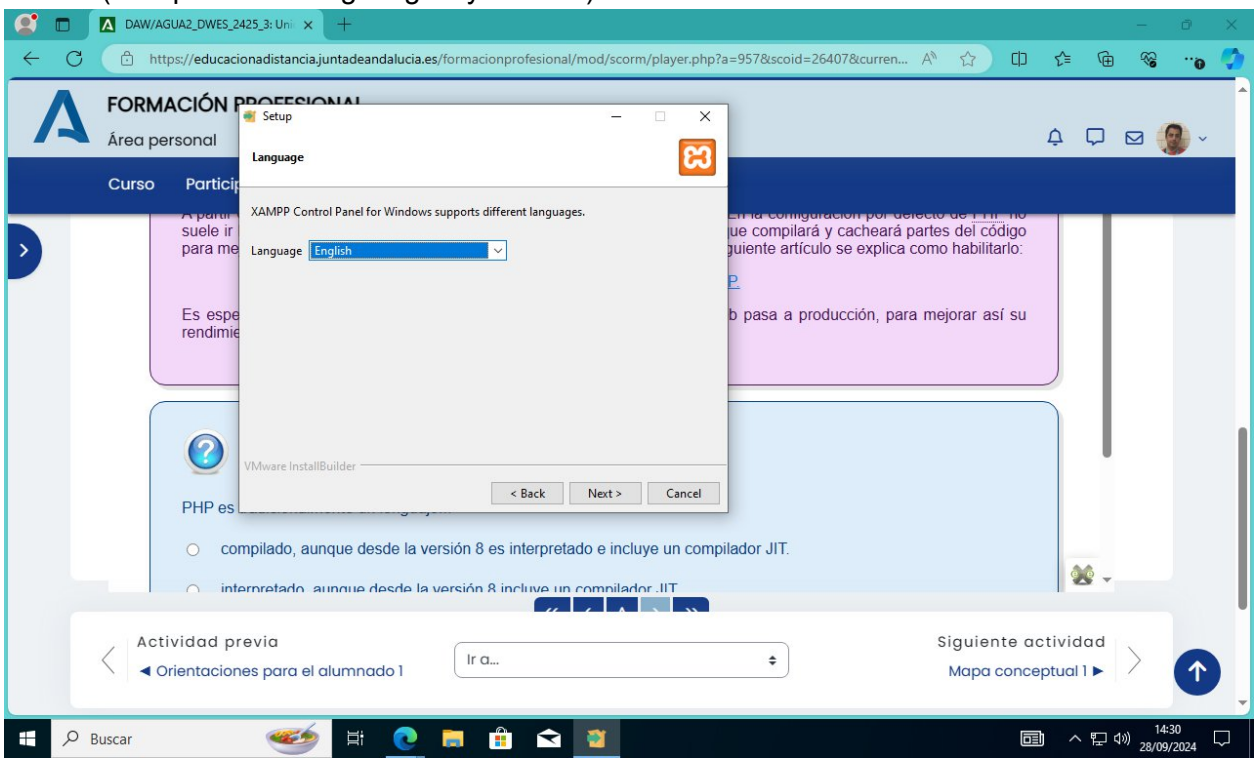
Podemos desmarcar todo lo que no sea necesario, aunque no es imprescindible, por lo que vamos a dejar lo básico (MySQL, Apache, PHP).



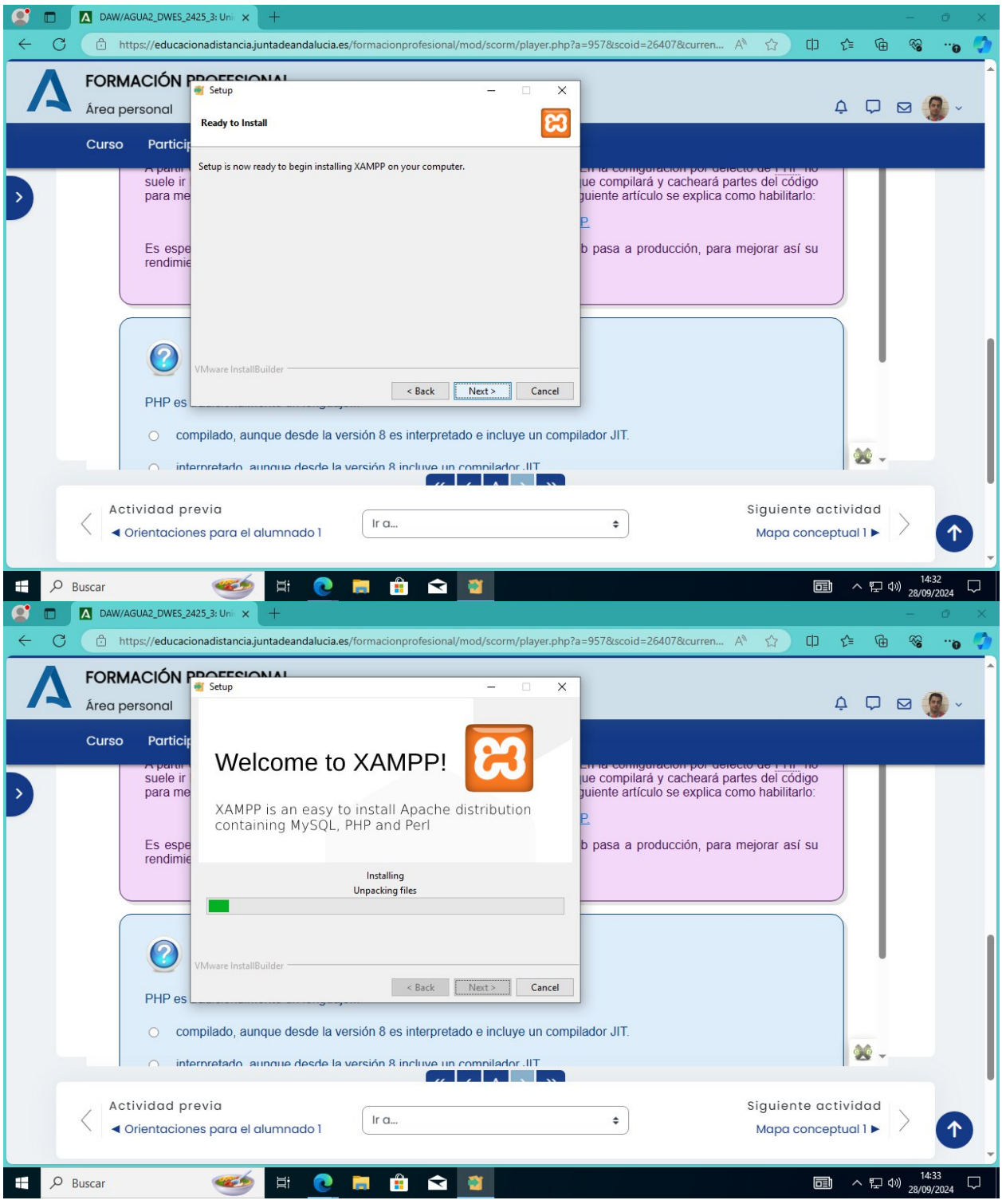
Carpeta donde se instalará

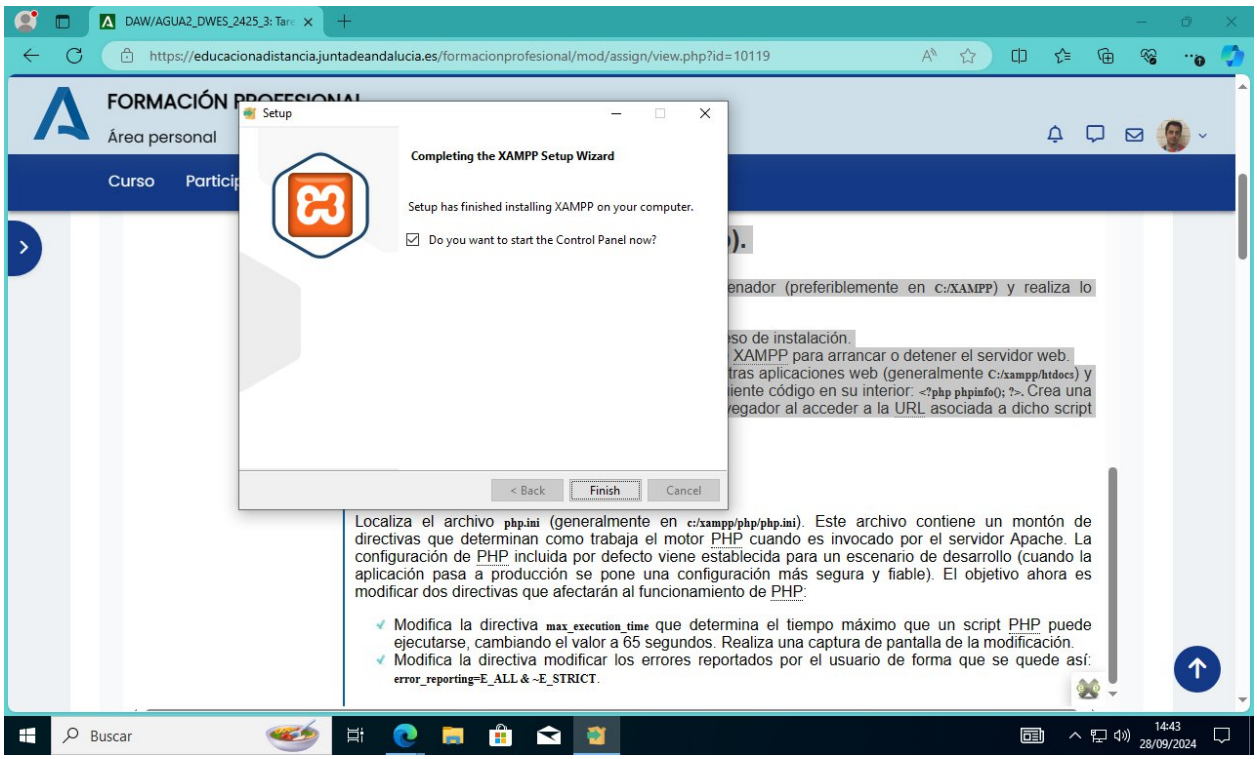


Idioma (sólo podemos elegir inglés y alemán)



Ya estaría todo preparado, pulsamos “Next” y esperamos que finalice la instalación.

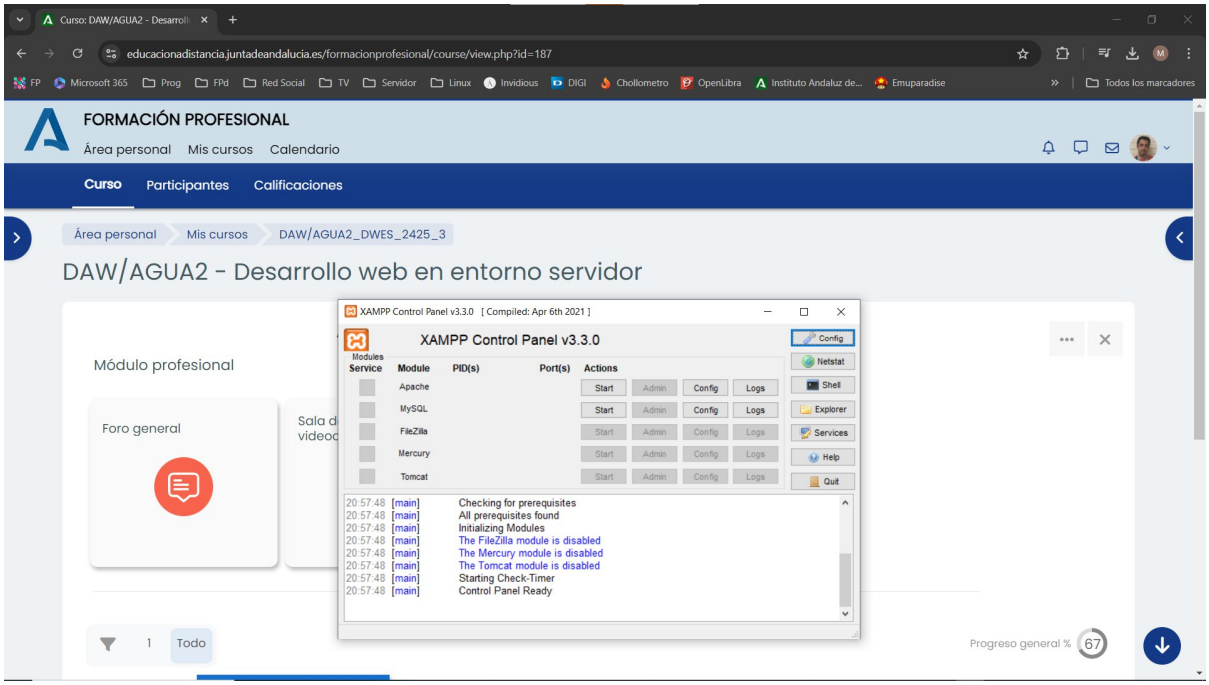

DAW/AGUA2_DWES_2425_3: Uni
https://educacionadistancia.juntadeandalucia.es/formacionprofesional/mod/scorm/player.php?a=957&scoid=26407&curren...
FORMACIÓN PROFESIONAL
Área personal
Curso
Setup
Ready to Install
Setup is now ready to begin installing XAMPP on your computer.
VMware InstallBuilder
< Back
Next >
Cancel
compilado, aunque desde la versión 8 es interpretado e incluye un compilador JIT.
Actividad previa
◄ Orientaciones para el alumnado 1
Ir a...
Siguiente actividad
Mapa conceptual 1 ►
Buscar
14:32
28/09/2024
Welcome to XAMPP!
XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl
Installing
Unpacking files
14:33



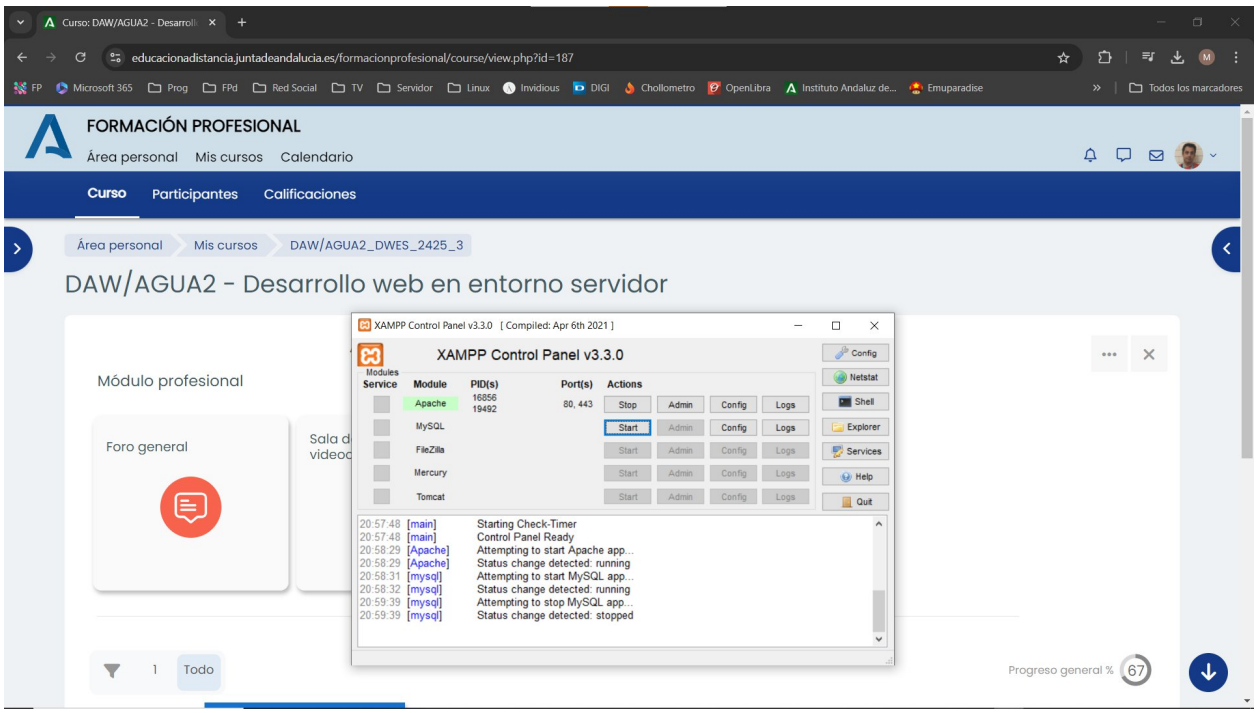
Una vez finalizada la instalación continuamos con la tarea.

- Realiza una captura de pantalla de la interfaz de XAMPP para arrancar o detener el servidor web.

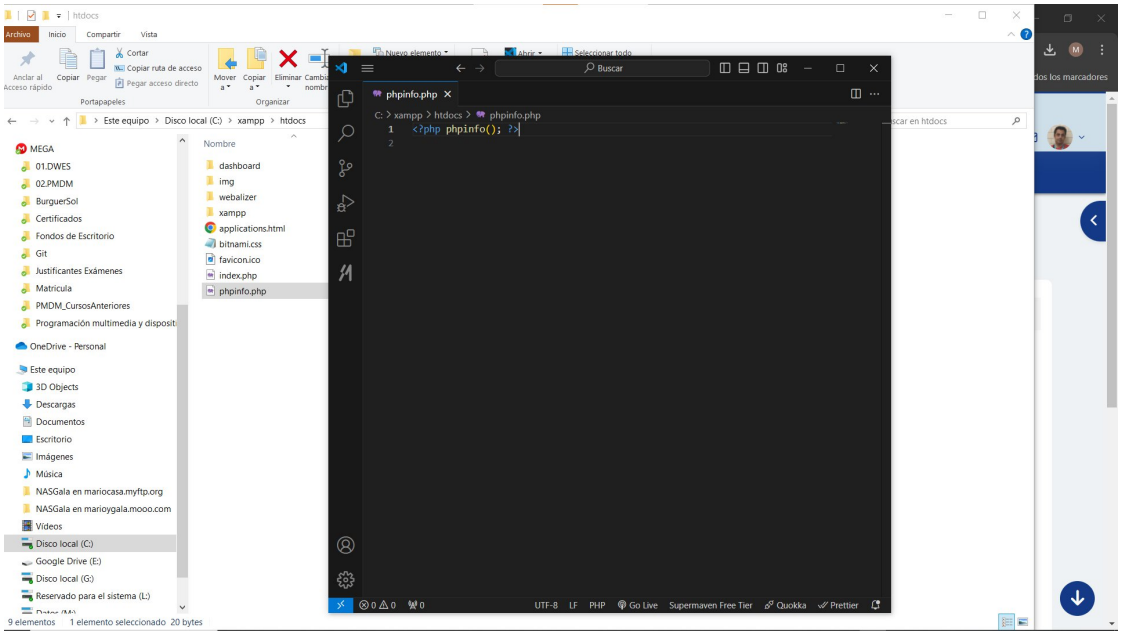
En esta primera captura vemos que podemos iniciar los servicios “Start”, configurar “Config” y ver los logs “Logs”.

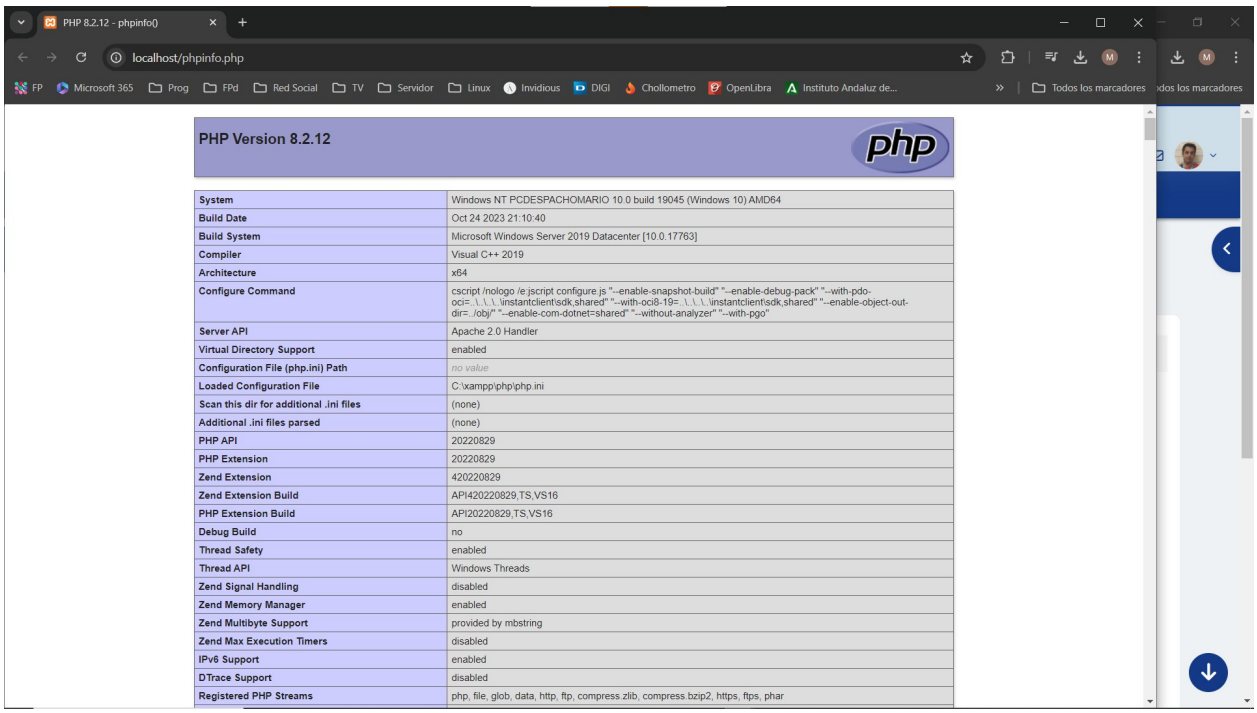


En esta captura como ya tenemos iniciado el servicio Apache, podemos parar “Stop”, administrar “Admin”, y el resto igual.



- Localiza la carpeta donde podemos añadir nuestras aplicaciones web (generalmente `C:/xampp/htdocs`) y añade un script llamado `phpinfo.php`. Añade el siguiente código en su interior: `<?php phpinfo(); ?>`. Crea una captura de pantalla de lo que aparece en el navegador al acceder a la URL asociada a dicho script (previsiblemente será http://localhost/phpinfo.php).





System	Windows NT PCDESPACHOMARIO 10.0 build 19045 (Windows 10) AMD64
Build Date	Oct 24 2023 21:10:40
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=..\..\..\instantclient\sdk,shared" "--with-oci8-19=..\..\..\instantclient\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829,TS,VS16
PHP Extension Build	API20220829,TS,VS16
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, compress.zlib, compress.bzip2, https, ftps, phar

Ejercicio 5 (RA1.d, RA2.f).

Localiza el archivo **`php.ini`** (generalmente en **`c:/xampp/php/php.ini`**). Este archivo contiene un montón de directivas que determinan como trabaja el motor PHP cuando es invocado por el servidor Apache. La configuración de PHP incluida por defecto viene establecida para un escenario de desarrollo (cuando la aplicación pasa a producción se pone una configuración más segura y fiable). El objetivo ahora es modificar dos directivas que afectarán al funcionamiento de PHP:

- Modifica la directiva **`max_execution_time`** que determina el tiempo máximo que un script PHP puede ejecutarse, cambiando el valor a 65 segundos. Realiza una captura de pantalla de la modificación.

- Modifica la directiva modificar los errores reportados por el usuario de forma que se quede así: **`error_reporting=E_ALL & ~E_STRICT`**.

1.2.- Sección 2.

Orientaciones para el alumnado

En este ejercicio se integran los criterios de evaluación siguientes:

- RA2.c) Se han utilizado etiquetas para la inclusión de código en el lenguaje de marcas.

- RA2.d) Se ha reconocido la sintaxis del lenguaje de programación que se ha de utilizar.

- RA2.e) Se han escrito sentencias simples y se han comprobado sus efectos en el documento resultante.
- RA2.g) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.
- RA2.h) Se han identificado los ámbitos de utilización de las variables.
- RA3.a) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.
- RA3.b) Se han utilizado bucles y se ha verificado su funcionamiento.
- RA3.c) Se han utilizado matrices (*arrays*) para almacenar y recuperar conjuntos de datos.
- RA3.d) Se han creado y utilizado funciones.

Importante: como parte de esta entrega el alumnado deberá aportar el código fuente y un vídeo de no más de 5 minutos explicando, con su voz propia, lo realizado en los ejercicios, las dificultades encontradas y un ejemplo de funcionamiento. Se aportará un vídeo en total para la sección 2 y 3.

Para realizar este ejercicio tienes que crear cinco archivos diferenciados: **`dwes01/ejercicio2/index.php`**, **`dwes01/ejercicio2/a.php`** y **`dwes01/ejercicio2/b.php`**; y dos archivos CSV (**`dwes01/ejercicio2/datos1.csv`** y **`dwes01/ejercicio2/datos2.csv`**).

Para empezar, diseña dos archivos CSV con el mismo número de columnas (al menos 4) y exactamente 5 filas en cada uno. El archivo **`datos2.csv`** debe ser la continuación de **`datos1.csv`**. Los archivos CSV son "datos separados por comas" y es una forma de almacenar datos tabulados (como una tabla en una hoja de cálculo). El contenido de los archivos CSV es a elegir por ti, pero deben de ser datos realistas que podría manejar una empresa o entidad. Puedes escribirlos en una hoja de cálculo (LibreOffice Calc por ejemplo) y exportar a CSV ambos archivos.

Sugerencias para el contenido de los archivos CSV: productos, discos, libros, participantes en un concurso o competición, partidos y resultados, datos de pluviometría, etc.

A continuación, en el archivo **`a.php`** realizarás una función destinada a cargar los archivos CSV (situados en **`dwes01/ejercicio2`**). Dicha función recibirá por parámetro un array indexado pasado por referencia y también el nombre del archivo CSV a cargar. La función deberá **añadir al array** pasado por referencia los datos cargados del archivo CSV. Esta función retornará **`true`** si se pudo abrir el archivo CSV y cargar su contenido al array pasado por referencia, o **`false`** en caso contrario. Es necesario que los datos del archivo CSV se carguen usando **`fgetcsv`**.

En el archivo **`b.php`**, por otra parte, realizarás una función que tendrá uno o más parámetros, que use la función creada en **`a.php`** y que retorne un array usando **`return`**. Esta función cargará los dos archivos CSV de forma consecutiva usando la función

creada en **`a.php`**, **acumulando de forma adecuada** el contenido de los archivos CSV en un único array bidimensional. Después, **usará la información de los parámetros para filtrar las filas** en función de alguna columna (condición mayor que, menor que, igual a, valores dentro de un conjunto, etc.). Las condiciones de filtrado son de tu elección, es decir, tu eliges como y que filtros se implementan.

Ejemplos de filtrado: si los datos incluyen precios el filtrado podría ser por "precio mayor que", si los datos incluyen una columna "categoría" el filtrado podría ser "categoria igual a",…

En **`index.php`** deberás usar la función creada en **`b.php`** para mostrar una tabla HTML correctamente formateada con los datos cribados. La tabla HTML debe generarse obligatoriamente en el archivo **`index.php`** usando el array retornado por la función creada en **`b.php`**.

Por último, se valorará que se documente el código aportado, especialmente las funciones.

Criterios de corrección:

1. Todos los archivos tienen que tener tu nombre y apellidos al principio. El contenido y formato de los archivos CSV tiene que ser propio y diferente al de tus compañeros.
2. Los archivos se cargan entre sí de forma adecuada usando **`include`**, **`include_once`**, **`require`** o **`require_once`**.
3. Los archivos **`a.php`** y **`b.php`** contienen el código descrito (funciones) y no tienen código que no corresponda con la función descrita.
4. Las funciones creadas en **`a.php`** o **`b.php`** no generan texto ni HTML que se envíe al navegador.
5. La función creada en **`a.php`** debe usar obligatoriamente la función **`fgetcsv`** para cargar los datos.
6. La función creada en **`a.php`** debe cumplir con lo indicado en el enunciado (array pasado por referencia al que se añade el contenido leído del archivo CSV)
7. La función creada en **`b.php`** debe obligatoriamente usar la función creada en **`a.php`** dos veces de forma adecuada para cargar los archivos CSV.
8. La función creada en **`b.php`** debe realizar un filtrado de datos generando un nuevo array o eliminando filas de forma adecuada.
9. El archivo **`index.php`** muestra los datos de la forma indicada en el enunciado (tabla HTML) usando la función creada en **`b.php`** para obtener los datos a mostrar.
10. Las tablas generada en **`index.php`** están adecuadamente formateadas.
11. Se han documentado las funciones y el código creado.

12. Variaciones en los datos suministrados a las funciones NO DEBEN GENERAR ERRORES.

1.3.- Sección 3.

Orientaciones para el alumnado

En esta tarea se integran lo siguientes criterios de evaluación:

- RA2.g) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.
- RA3.a) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.
- RA3.b) Se han utilizado bucles y se ha verificado su funcionamiento.
- RA3.c) Se han utilizado matrices (*arrays*) para almacenar y recuperar conjuntos de datos.
- RA3.e) Se han utilizado formularios Web para interactuar con el usuario del navegador Web.
- RA3.f) Se han empleado métodos para recuperar la información introducida en el formulario.
- RA3.g) Se han añadido comentarios al código.

Importante: como parte de esta entrega el alumnado deberá aportar el código fuente y un vídeo de no más de 5 minutos explicando, con su voz propia, lo realizado en los ejercicios, las dificultades encontradas y un ejemplo de funcionamiento. Se aportará un vídeo en total para la sección 2 y 3.

En esta sección deberás crear dos archivos, uno llamado **`dwes01/ejercicio3/index.php`** en el que aparezca un formulario con el que se pretende buscar datos dentro de un conjunto de datos (*array* bidimensional), y otro llamado **`dwes01/ejercicio3/datos.php`** para mostrar los datos filtrados de dicho conjunto de datos. **No se usarán funciones en este caso.**

En el archivo **`index.php`** debe explicarse con texto (HTML visible) el funcionamiento del formulario. El formulario del archivo **`index.php`** deberá tener obligatoriamente como mínimo los siguientes campos:

- Un campo tipo **`type='text'`** destinado a almacenar un dato numérico (**NO** puede usarse **`type='number'`**).
- Al menos cuatro selectores tipo **`type='checkbox'`** o bien **`<select multiple>`** que se reciban como un *array* (no como datos individuales), donde el usuario podrá seleccionar uno o más elementos.

Puedes incluir más campos en el formulario, pero como mínimo deben estar los anteriores. Si no se incluyen, el ejercicio no se considerará por realizado.

El destino del formulario será el archivo **`datos.php`** y los datos e enviarán usando el método **`POST`**.

Los datos a filtrar estarán almacenados en un *array* bidimensional en PHP (primera dimensión indexada, segunda dimensión asociativa) en el mismo archivo **`datos.php`**, teniendo en cuenta que puede haber uno o más registros que puedan potencialmente cumplir con las condiciones de filtrado.

El *script* **`datos.php`** debe mostrar una tabla HTML adecuadamente formateada con los datos filtrados en base a los datos recibidos del formulario, procesando los datos recibidos de forma adecuada acorde a cada tipo de campo y contenido, y evitando cualquier error producido por la no existencia de dichos datos. El resultado del filtrado debe poder mostrar, en base a los datos recibidos, una o más filas/registros acorde a los datos recibidos vía POST.

En caso de que algún dato recibido por `datos.php` no sea el esperado (letras en vez de un número, un valor fuera del rango de valores posibles en la selección múltiple o select, etc.), se mostrarán errores acordes al problema encontrado.

En caso de que el *script* `datos.php` no reciba ningún dato debe mostrar el formulario incluyendo el archivo `index.php` (generando un HTML correcto). En este caso, debes diferenciar el envío de datos vacíos del hecho de "no enviar datos". En el primer caso se envía el formulario sin rellenar nada, y en el segundo caso, no se llega a enviar el formulario (accediendo por ejemplo directamente a `datos.php`).

Por último, se valorará que se documente el código aportado de forma apropiada..

Criterios de corrección:

1. Todos los archivos tienen que tener tu nombre y apellidos al principio. El contenido y formato de los datos y formularios tiene que ser propio y diferente al de tus compañeros.

2. El formulario tiene que usar de forma obligatoria uno de los campos de tipo múltiple, donde los datos se recibirán como un *array*. El otro será, de forma obligatoria, tipo **`text`** aunque se usará para enviar un dato numérico.

3. El formulario tiene que ser obligatoriamente tipo POST y el **`action`** se configurará de forma adecuada.

4. Los datos recibidos en el *array* **`$_POST`** se validarán y verificarán de forma adecuada en **`datos.php`**, sin producir errores por la no existencia de datos o por datos no esperados. En el caso de los datos múltiples se debe verificar siempre que el conjunto de datos recibidos esté dentro del conjunto de valores posibles. En caso del dato tipo **`text`** deberá verificarse obligatorimente que es un número.

5. En **`datos.php`** se muestran errores detallados en caso de que los datos recibidos no cumplan con los criterios de validación y verificación especificados.

6. En **`datos.php`** los datos mostrados incluirán todas las filas que cumplan con las condiciones indicadas en el formulario, existiendo siempre casos donde se muestren una o más filas.

7. En **`datos.php`** cuando no se reciben los datos **`$_POST`** se mostrará el formulario incorporando **`index.php`** de forma adecuada para generar un HTML correcto.

8. Se han documentado las funciones y el código creado.