UNIDAD 3 – PHP

1. UNIDAD 3: Introducción

Bienvenido a Backend

Estudiaremos uno de los lenguajes de programación del lado del servidor más popular y utilizado en la Web: PHP. Un lenguaje interpretado y de alto rendimiento que es utilizado para escribir servidores webs y realizar conexiones a bases de datos.

 Puntos de aprendizaje

Unidad 3: PHP  
¡Aquí comienza la Unidad 3: PHP!

* Los objetivos del Programa que corresponden a esta Unidad son:
  + Usar metodologías sincrónicas y asincrónicas para gestionar datos desde el servidor.
* Los objetivos del Curso 6 que corresponden a esta Unidad son:
  + Describir el funcionamiento de la ejecución sincrónica y asincrónica
  + Definir qué es una API y cómo funcionan los Servicios Web
  + Implementar AJAX
  + Comprender qué es y cómo funciona PHP

1. Lección 1: Introducción a PHP
   1. ¿Qué es PHP?

PHP es el acrónimo recursivo de *Hypertext Protocol* . Es un lenguaje de programación que se ejecuta del lado del servidor y se usa especialmente para el desarrollo de proyectos web. PHP es un lenguaje muy versátil y puede ser usado en múltiples aplicaciones. En su uso más común es un lenguaje interpretado, es decir, que sus instrucciones se traducen una a una en tiempo de ejecución; sin embargo, si se usa para aplicaciones de escritorio, apoyado en motores adicionales, puede convertirse en un lenguaje compilado.  
  
Una de las características principales de PHP es que es un lenguaje que se puede mezclar con HTML para realizar procesos lógicos en el mismo archivo que muestra información en el navegador. Esta ventaja permite realizar proyectos que interactúan con bases de datos, formularios que envían información, verificar usuarios para el acceso a sitios y muchas otras posibilidades adicionales.  
  
PHP es un lenguaje de código abierto, es decir, su descarga es gratuita y además cuenta con una comunidad extensa de desarrolladores que aportan al mejoramiento y estabilidad del mismo. La versión actual es PHP 7, que fue lanzada el 3 de diciembre de 2015.

* 1. ¿Qué puede hacer PHP?

 Infografía ¿Qué puedo hacer con PHP?

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. A continuación, hemos diseñado un recurso para ti sobre las bondades de PHP.

* 1. Instalación

Servidor XAMPP 🡪 Sirve para simular el lado del servidor

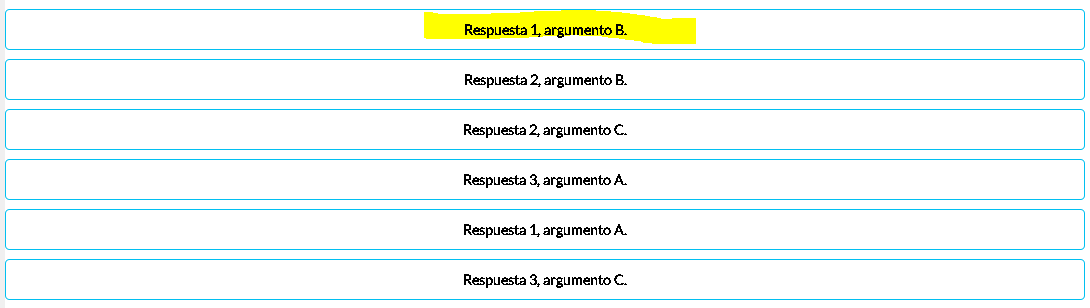
* 1. Iniciar con PHP
  2. Actividad

Tiempo de pensar

En esta lección pudiste dar tus primeros pasos con PHP, un lenguaje de programación secuencial que permite realizar procesos extraordinarios del lado del servidor para crear sitios web funcionales y poderosos. Adicionalmente, viste paso a paso la configuración inicial del entorno para trabajar con este lenguaje.  
  
Instrucciones:  
  
Lee la pregunta a continuación y observa las tres posibles respuestas. Adicionalmente se presentan una serie de afirmaciones y argumentos que las soportan. Elige la opción que reúna la afirmación y el argumento correcto a la pregunta propuesta.  
  
Pregunta:  
  
¿PHP es un lenguaje interpretado o compilado?  
  
Afirmaciones:

1. Normalmente interpretado, pero en ocasiones compilado.
2. Normalmente compilado, pero en ocasiones interpretado.
3. Interpretado y Compilado en igual nivel.

Argumentos:  
  
  A. Porque sus instrucciones se traducen en tiempo de ejecución o tratadas por un compilador que las traduce a   lenguaje de máquina.  
  B. Porque en su uso convencional sus instrucciones se traducen en tiempo de ejecución, pero con la ayuda de herramientas pueden hacer uso de un compilador y ser traducidas a lenguaje de máquina.  
  C. Porque en su uso convencional sus instrucciones hacen uso de un compilador y son traducidas a lenguaje de máquina, pero con la ayuda de herramientas pueden ser traducidas en tiempo de ejecución.



1. Lección 2: Sintaxis y estructuras de control
   1. Sintáxis Básica
   2. Variables
   3. Constantes

Es una representación de un valor fijo. El valor no puede cambiar:

2 formas:

* Const 🡪 Constantes de una propiedad de una clase se usa const
* Función define() 🡪 Si definimos una constante con un tipo de dato diferente se utiliza define()
  + Ej: define(‘A’)

Cuando se crean las constantes no se pueden eliminar o modificar. Y las constantes pueden ser accedidas desde cualquier ámbito.

Constantes mágicas, el valor puede variar: LINE

* FILE
* DIR
* FUNCTION
* CLASS
* METHOD
* TRAIT
* NAMESPACE
  1. Operadores y Condicionales

Operadores 🡪 nos permiten interactuar con variables y constantes

Ej: $a += 2

Operadores aritméticos:

* Suma
* Resta
* Div
* Mult
* Potenciación $a \*\* $b
* Residuo $a % $b

Condicionales básicos:

* $a == $b
* $a ¡= $b
* $a > $b
* $a < $b
* $a >= $b
* $a <= $b
* $a === $b
* $a ¡== $b
* $a <=> $b 🡪 retorna -1 si b es mayor que a, 0 si son iguales y 1 si a es mayor que b
* $a??$b??$c 🡪 operador de fusión, retorna el primer operando no nulo que exista en la expresión
* $a && $b 🡪 Operadores lógicos
* $a || $b
* $a xor $b 🡪 true (si alguno es true)
* ¡$a 🡪 Si $a es true. Operador not

Condicionales:

If (condicion){

Accion1

}elseif(condicion){

Accion3

}

else{

Accion2

}

* 1. Ciclos

While (condicion){

accion

}

Do {

accion

} while (condicion)

For (valor\_inical; condicion;contador){

accion

}

Foreach($arreglo as $posicion => $valor){

acciones

}

* 1. Actividad

Tiempo de pensar

Es muy importante que tengas claros los fundamentos básicos de la programación en PHP, para que sea más fácil dominarlo en totalidad. La definición de variables y constantes, así como de los condicionales y ciclos, es muy importante ya que deberás usarlos con mucha frecuencia al programar en este lenguaje.  
  
Instrucciones:  
  
Observa el código que se te presenta a continuación, en donde los contenedores con la clase cuadroNegro y cuadroBlanco son bloques cuyo fondo es del color que indica su nombre y se ubican uno al lado del otro horizontalmente. De acuerdo con el código, elige la afirmación verdadera.

Código: (php y html)

#### PHP

**<div class="tablero">**

**<?php**

**$i = 1;**

**while ($i < 5) {**

**if ($i%2==0) {**

**?>**

**<div class="cuadroNegro"></div>**

**<?php**

**}else {**

**?>**

**<div class="cuadroBlanco"></div>**

**<?php**

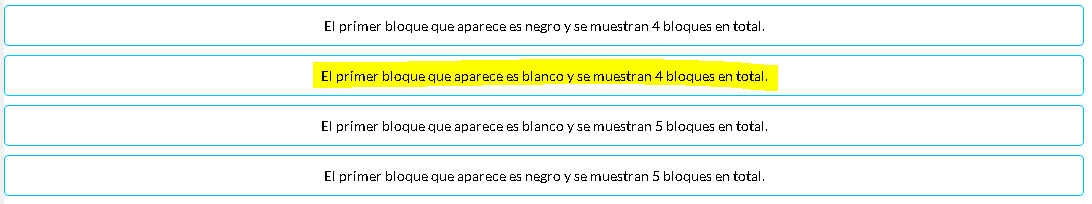
**}**

**$i+=1;**

**}**

**?>**

**</div>**



* 1. Ejercicio de Codificación

HORA DE  
codificar

PHP es un lenguaje de programación que le otorga el cerebro a nuestros proyectos web. Gracias a sus características ahora no sólo podemos crear sitios web estéticos y llamativos sino también funcionales. En su sintaxis básica encontramos el uso del signo pesos ($) para la denominación de variables y la forma de definir constantes, condicionales y estructuras iterativas.  
  
En este ejercicio debes usar las estructuras de control, ciclos, variables y constantes de PHP para mostrar el contenido adecuado a cada uno de los roles de un sistema de gestión académico. Adicionalmente debes usar tus conocimientos para listar los grados de la institución educativa y construir un calendario.



**Contenido**

Para el desarrollo de este ejercicio debes descargar el archivo base llamado codBase.zip. En él encontrarás una página HTML con las bases gráficas del sistema. A partir de los elementos que allí se encuentran, debes seguir las siguientes instrucciones para completar el ejercicio.

**Instrucciones**

Cambia la extensión del archivo a la adecuada para que sea posible insertar código PHP en él.

Crea un arreglo de constantes por cada uno de los siguientes listados. Para hacerlo debes asignar el valor a la constante creada una función array y entre paréntesis los elementos del arreglo. Ejemplo:



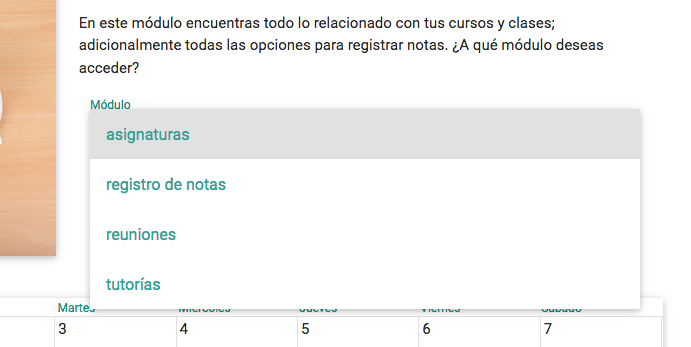
* Categorías: Profesor, Estudiante, Padre.
* Módulos Profesor: Asignaturas, registro de notas, reuniones, tutorías.
* Módulos Estudiante: Asignaturas cursadas, registro de notas.
* Módulos Padre: Reuniones, registro de notas.

Crea una variable que almacene el tipo de usuario que accede a la página, su valor puede ser profesor, estudiante o padre.

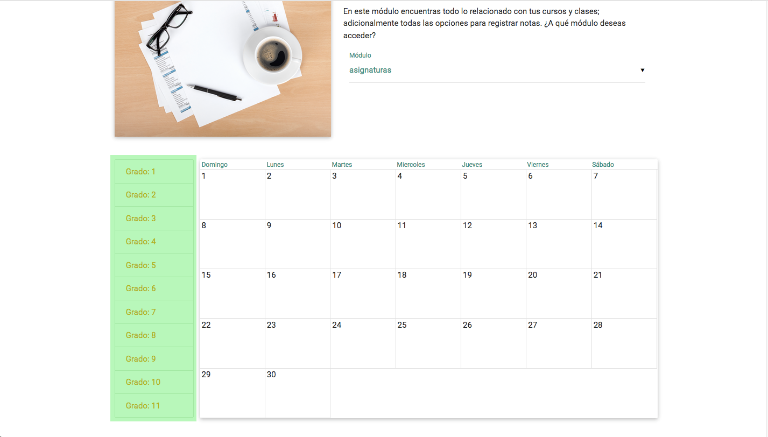
Muestra el contenedor de la clase mod-container correspondiente de acuerdo al valor de la variable definida en el paso anterior. Esto quiere decir que si la variable tiene el valor ‘profesor’ sólo se debe mostrar en la página el contenedor que tenga el contenido del profesor. La comparación de la variable se debe hacer con el arreglo de constantes Categorías.



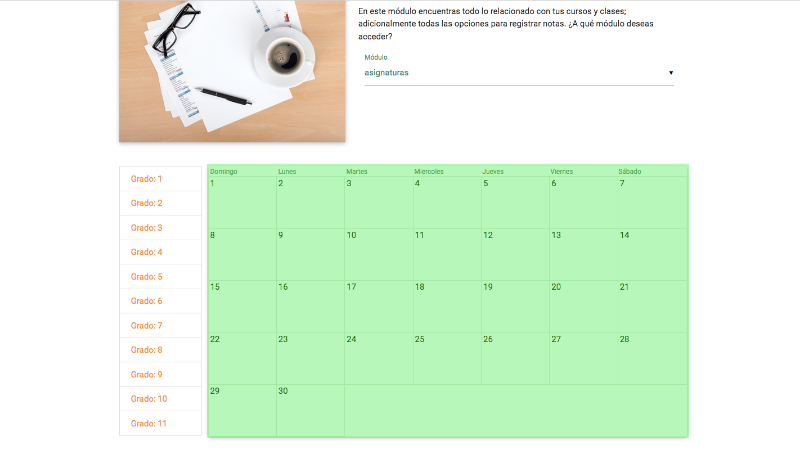
Crea por medio de un ciclo los elementos option del select correspondientes a los módulos de cada tipo de usuario, recorriendo el arreglo de constantes correspondiente a cada categoría.



Llena la lista de Grados en la parte inferior izquierda usando un ciclo diferente al usado en pasos anteriores. Los grados deben mostrarse del 1 al 11.

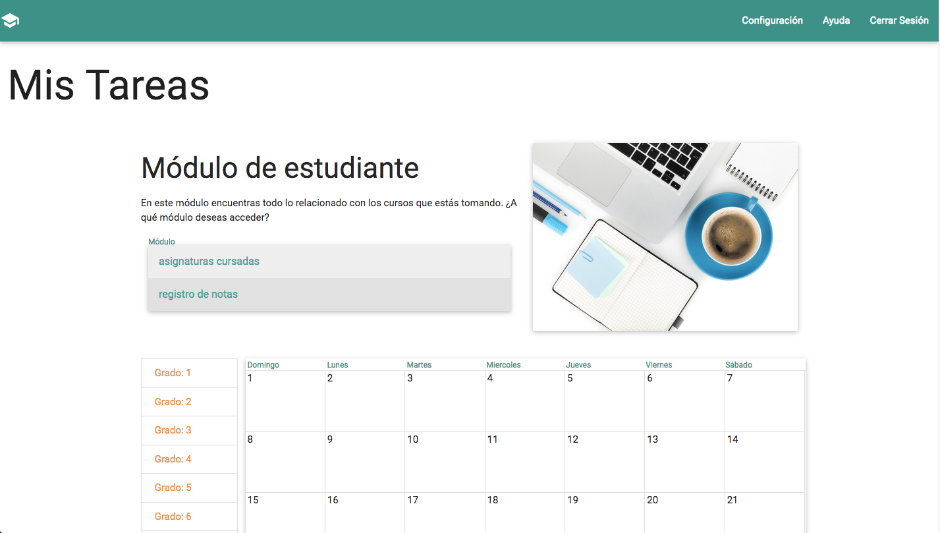


Crea los días del calendario de 1 a 30 usando un ciclo diferente a los usados en pasos anteriores.



Comprueba el correcto funcionamiento de la página cambiando el valor de la variable que indica el tipo de usuario que accede al sistema.







Buenas Prácticas

El desarrollo con PHP inmerso dentro del HTML puede ser muy riesgoso desde el punto de vista de la estructura básica de ambos lenguajes. Siempre debes fijarte que las etiquetas de apertura y cierre estén ubicadas correctamente, de lo contrario es probable que existan errores en tu código que en ocasiones son indetectables y corre peligro el buen rendimiento de tu sitio web.

1. Lección 3: Funciones y Objetos
   1. Funciones

Function saltar () {

Acciones de la función

}

Las funciones son globales. Puede Retornar un resultado

Function saltar () {

Acciones de la función

Return $resultado

}

* 1. Clases e instancias
* Clases 🡪 POO
* Instancias

Class NombreClase{

Propiedades

Métodos

}

* Visibilidad – Ambitos desde donde puede ser accedidos los métodos y propiedades de una clase
  + Public 🡪 Si el atributo o método puede ser accedido por fuera de la clase
  + Protected 🡪 Si el atributo o método solo puede ser accedido desde la misma clase o mediante clases heredadas
  + Private 🡪 Si el atributo o método solo puede ser accedido desde la misma clase que lo definió.

Propiedad Estática:

Solo puede ser accedido mediante la clase como tal y no mediante una instancia de la misma

Const son propiedades estáticas por defecto

Constructor:

Es el método de una clase que realiza una serie de acciones predefinidas cada vez que se instancia un nuevo objeto de la clase correspondiente.

Class nombreClass{

Private $atributo;

Function \_\_construct($argumento){

$atributo = $argumento;

}

}

$obj = new nombreClass(“valor”)



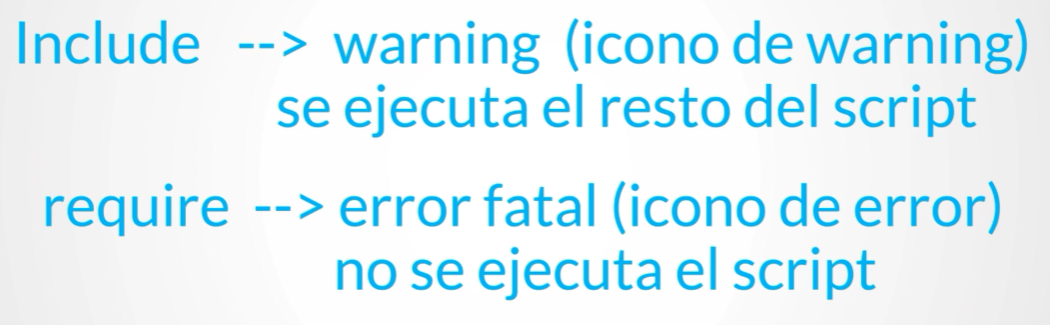
Se usa con frecuencia para iniciar propiedades en los constructores

* 1. Inclusión de archivos

Se incluyen archivos externos mediante la palabra reservada include

La palabra reservada require 🡪 se utiliza para indicar inclusión obligatoria

Si el archivo a incluir no existe:



* 1. Excepciones

Crear advertencias personalizadas, en caso de errores.

Try {

Código a ejecutar

}

catch(Exception $e){

Acciones al lanzarse la excepción

} Finally{ // Este es opcional

Acciones que se van a ejecutar siempre, sin importar si se lanzan o no las exepciones

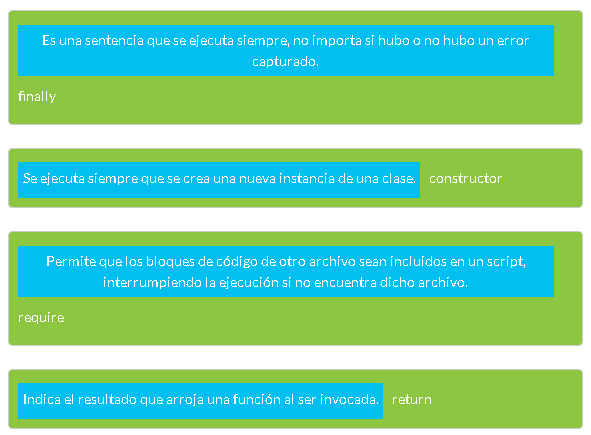
}

Ejemplo:

* 1. Actividad

Tiempo de pensar

En esta lección aprendiste uno de los conceptos primordiales en PHP: Las clases. Recuerda que el uso de clases y sus instancias corresponde al paradigma de programación orientada a objetos, la cual es ampliamente usada por los desarrolladores a nivel global. Además, el concepto de excepciones para administrar adecuadamente los posibles errores en los scripts y la inclusión de archivos para modularizar tus proyectos web, representan herramientas muy valiosas para la creación de sistemas robustos y bien estructurados.  
Instrucciones:  
  
A continuación verás dos columnas. En la columna de la izquierda encontrarás algunas definiciones o explicaciones de determinados conceptos de PHP y en la derecha algunos conceptos puntuales de este lenguaje. Asocia las definiciones en la izquierda con el concepto de PHP correspondiente en la columna derecha.



* 1. Ejercicio de codificación

HORA DE  
codificar

La programación orientada a objetos es uno de los paradigmas de programación más utilizados en el desarrollo de software. Los conceptos de clases, instancias y funciones representan una manera organizada y efectiva de estructurar grandes bloques de código, adicionalmente otorgan seguridad al acceso de datos encapsulándolos y generando un ambiente de desarrollo más limpio y mantenible.  
  
En PHP las funciones se definen mediante la palabra reservada “function” seguida de su nombre y los parámetros que deba recibir en su ejecución, por último, se indica el conjunto de acciones que realiza. Por otro lado, una clase se define mediante la palabra reservada “Class” seguida de su nombre y dentro de las llaves que la siguen se definen las propiedades y métodos que la componen.  
  
En este ejercicio continuarás desarrollando el sistema académico que iniciaste en el ejercicio de la lección anterior, añadiendo las funcionalidades de la página de asignaturas, en la que se debe mostrar información académica de acuerdo con el perfil que ha ingresado al sistema.

**Contenido**

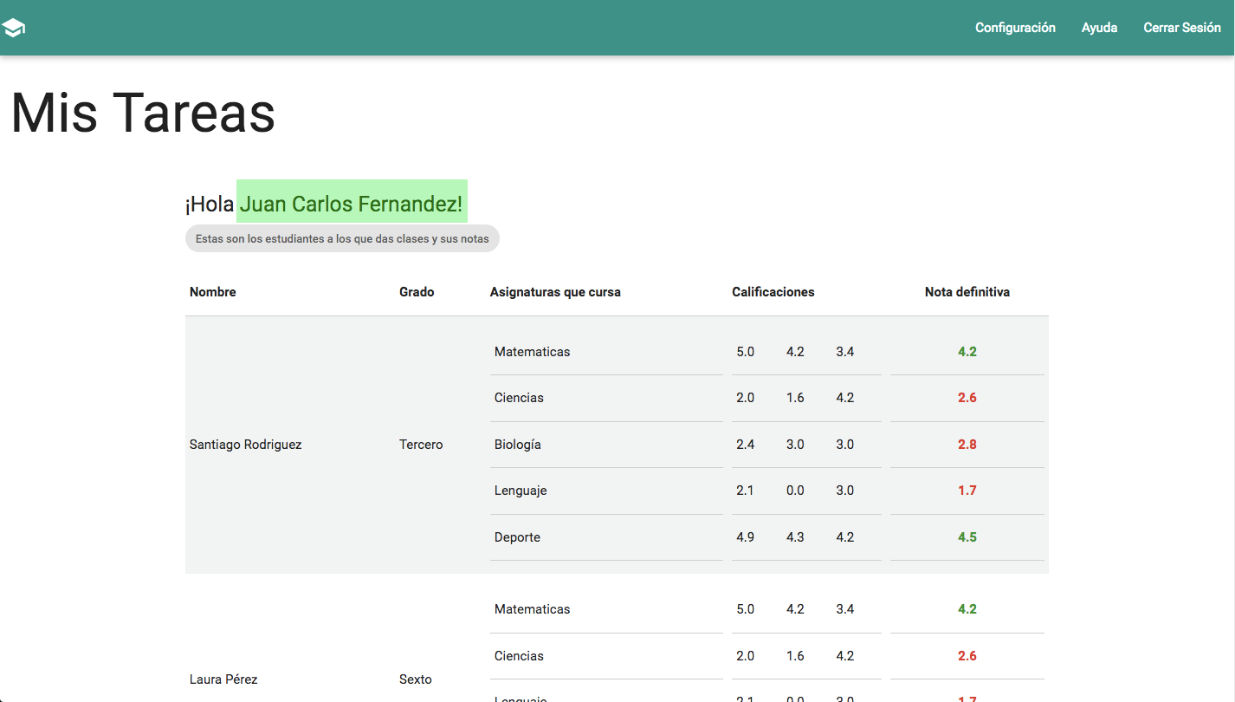
Para el desarrollo de este ejercicio debes continuar con la solución del ejercicio de la lección anterior.

**Instrucciones**

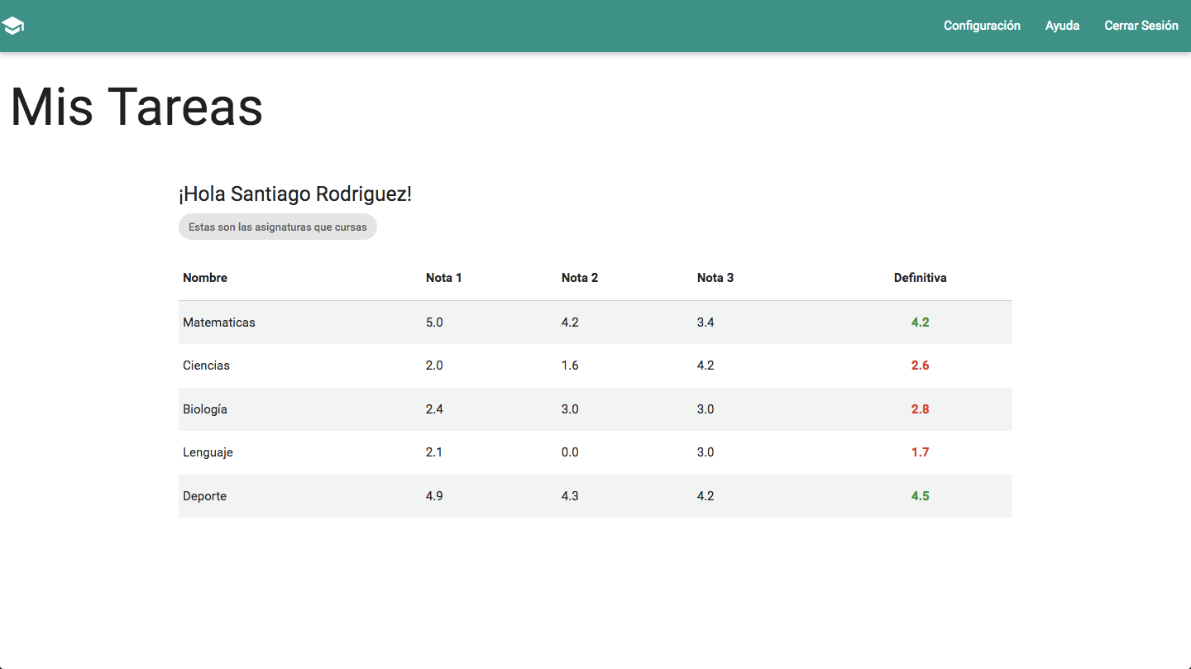
* Crea un nuevo archivo con el nombre “clases.php”
* Crea la clase Asignatura en el archivo creado, con las siguientes propiedades privadas: nombre, nota1, nota2 y nota 3. Para acceder a los valores de dichas propiedades debes crear dos métodos públicos por cada propiedad, uno que obtenga el valor de la propiedad y otro que le asigne un nuevo valor. La clase debe contar con su constructor correspondiente.
* Añade a la clase Asignatura un método que calcule el promedio de las 3 propiedades de del objeto que almacenan notas.
* Crea la clase Estudiante en el archivo creado, con las siguientes propiedades privadas: nombre, curso y un arreglo de asignaturas. Para acceder a los valores de dichas propiedades debes crear dos métodos públicos por cada propiedad, uno que obtenga el valor de la propiedad y otro que le asigne un nuevo valor. Para el arreglo de asignaturas sólo crea el método que retorne su valor. La clase debe contar con su constructor correspondiente.
* Añade a la clase Estudiante un método para adicionar un objeto de la clase Asignatura a la propiedad arreglo de asignaturas. Para esto debes especificar en los argumentos del método el tipo de dato que debe recibir, en este caso Asignatura y el nombre del parámetro. Para añadir el elemento ingresado como parámetro usa la función “array\_push” y en sus argumentos define primero el arreglo al cual será añadido el objeto, y el segundo parámetro indica el elemento a añadir. Ej:



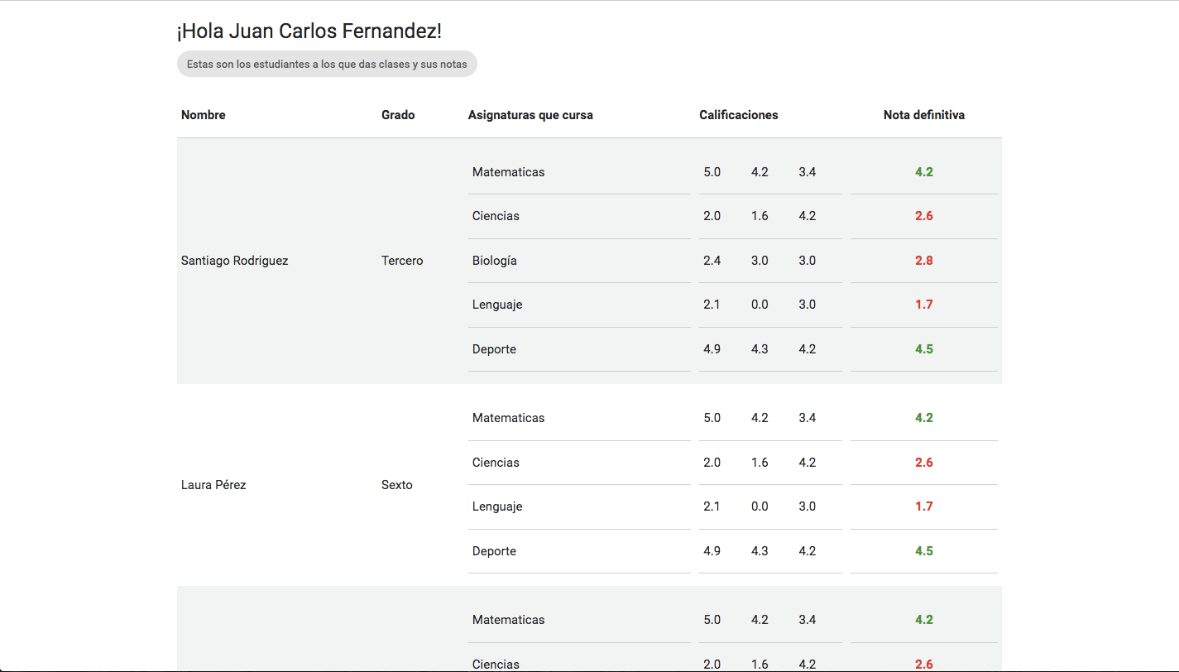
* Crea la clase Profesor en el archivo creado, con las siguientes propiedades privadas: nombre y un arreglo de estudiantes. Para acceder a los valores de dichas propiedades debes crear dos métodos públicos por cada propiedad, uno que obtenga el valor de la propiedad y otro que le asigne un nuevo valor. Para el arreglo de estudiantes sólo crea el método que retorne su valor. La clase debe contar con su constructor correspondiente.
* Añade a la clase Profesor un método para adicionar un objeto de la clase Estudiante a la propiedad arreglo de estudiantes.
* Crea la clase Padre en el archivo creado, con las siguientes propiedades privadas: nombre y un arreglo de hijos. Para acceder a los valores de dichas propiedades debes crear dos métodos públicos por cada propiedad, uno que obtenga el valor de la propiedad y otro que le asigne un nuevo valor. Para el arreglo de hijos sólo crea el método que retorne su valor. La clase debe contar con su constructor correspondiente.
* Añade a la clase Padre un método para adicionar un objeto de la clase Estudiante a la propiedad arreglo de hijos.
* Crea un nuevo archivo con el nombre “crear\_datos.php”. Allí debes incluir el archivo “clases.php”; si este archivo no se añade, el script no debe funcionar. En este archivo crea:
  + 5 instancias de la clase Asignatura, especificando todas sus propiedades.
  + 3 instancias de la clase Estudiante, añadiendo la cantidad de asignaturas que desees en su propiedad arreglo de asignaturas, y especificando sus demás propiedades.
  + 1 instancia de la clase Profesor, añadiendo la cantidad total de estudiantes creados, en su propiedad arreglo de asignaturas, y especificando sus demás propiedades.
  + 1 instancia de la clase Padre, añadiendo la cantidad total de estudiantes creados, en su propiedad arreglo de asignaturas, y especificando sus demás propiedades.
* Abre el archivo “registro.php” y en él incluye el archivo “crear\_datos.php”; si no se añade, el script puede ejecutarse de todas formas. Crea una variable con el nombre “usuario\_acceso” y asígnale una de las instancias de Estudiante, Profesor o Padre, creadas en pasos anteriores. Adicionalmente, crea una variable con el nombre “tipo\_acceso” que almacene el nombre de la clase del objeto que almacena la “usuario\_acceso”, usando la función “get\_class” y como argumento dicha variable.
* 
* Crea un condicional que muestre el bloque de código HTML correspondiente a cada perfil. Guíate por los comentarios en el archivo para identificar dichos bloques y a qué perfil pertenecen. Usa la variable “tipo\_acceso” para esta verificación.
* Muestra en el saludo inicial el nombre asociado al objeto definido en “usuario\_acceso”.  
  Ejemplo: desde un objeto de la clase Profesor.



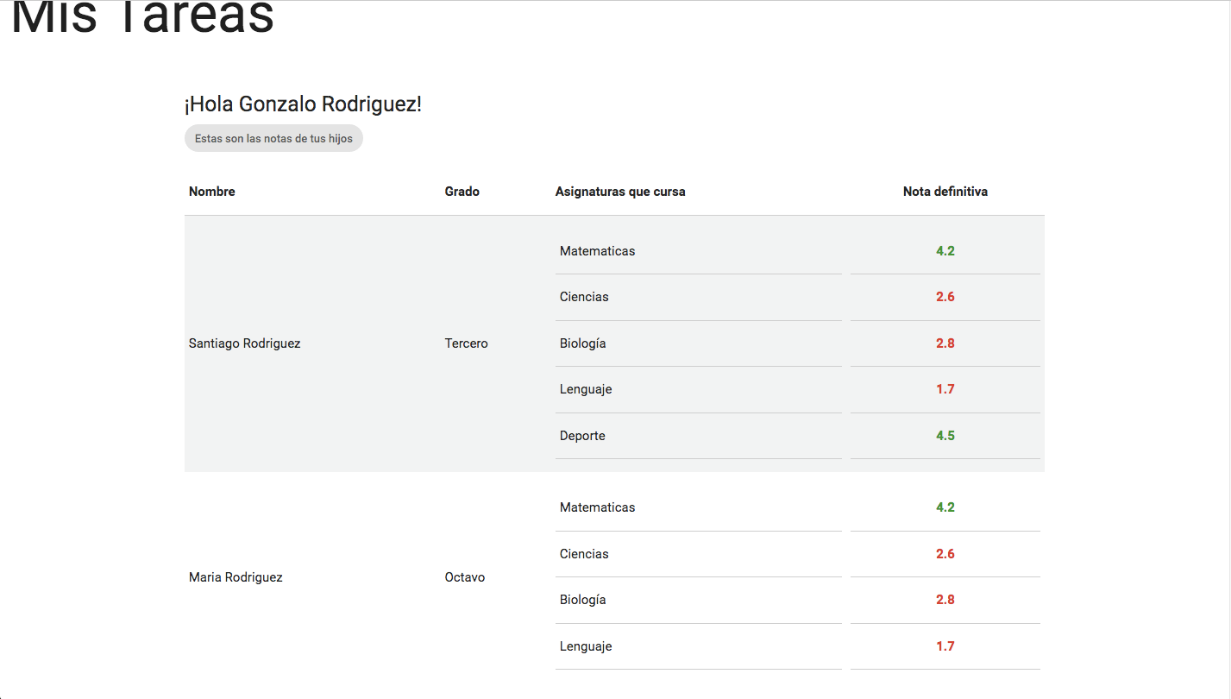
* Llena las tablas con la información correspondiente al objeto “usuario\_acceso”. Si es de tipo estudiante, listar las asignaturas que cursa, las notas correspondientes y el promedio.



* Llena las tablas con la información correspondiente al objeto “usuario\_acceso”. Si es de tipo profesor, listar los estudiantes a los que les da clase, el curso en el que se encuentran, las asignaturas que cursa cada uno de ellos, las notas correspondientes a dichas asignaturas y su promedio.



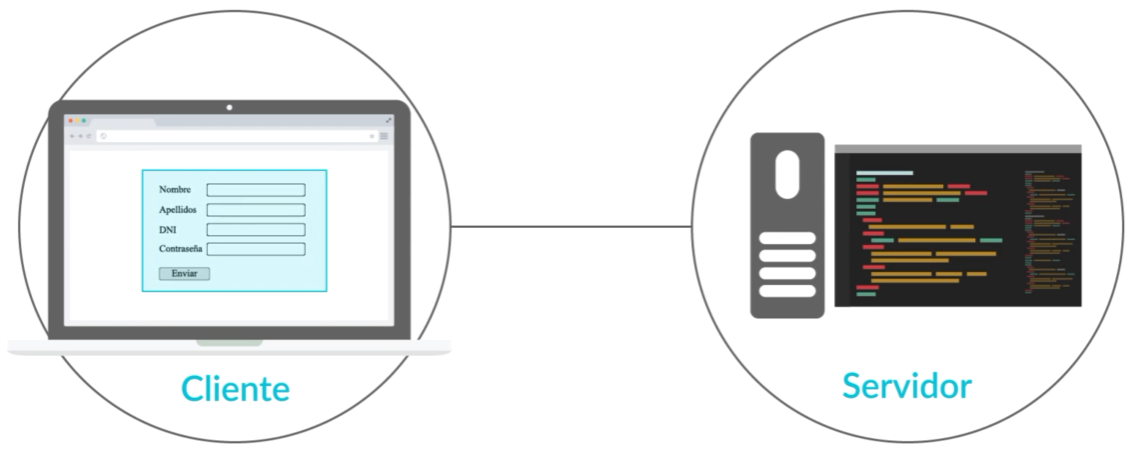
* Llena las tablas con la información correspondiente al objeto “usuario\_acceso”. Si es de tipo padre, listar sus hijos, el curso en el que se encuentran, las asignaturas que cursa cada uno de ellos y su promedio.



Buenas Prácticas

El uso adecuado de funciones y objetos permiten, además de un código organizado y mantenible, la reutilización de código; evitando así que repitas sentencias que hacen las mismas acciones y generando archivos con grandes cantidades de líneas que pueden afectar el rendimiento de todo el sistema.

1. Lección 4: Formularios
   1. Gestión del formulario HTML



Consideraciones:

* Debemos asignar la propiedad name a cada uno de las entradas del formulario, para así obtener los valores de los campos
* Los valores de los campos name deben ser valores explícitos para identificarlos fácilmente.
* Para radio buttons, checkboxes o select, se debe asignar el propiedad value.
* En los form se debe añadir la propiedad action 🡪 que indica a donde se va a dirigir la información del formulario una vez enviado (aquí se agregara el archivo script php que procesara los datos). También se debe especificar la propiedad del form, llamada method, que indica la forma en que se enviarán los datos del cliente hacia el servidor
  1. Recepción y envío de datos entre archivos

Métodos GET y POST para el envío de datos de un formulario

GET – Se encarga de llevar los datos desde el formulario hasta el servidor por medio de la URL. Estos se almacenas en php en un arreglo asociativo llamado $\_GET, Ej:

$\_GET[‘nombre\_del\_campo’] y este retorna el valor

POST – Envío de datos al servidor pero su funcionamiento es diferente a GET, ya que se mandan de forma oculta y no se ve en la URL. 🡪 Usado para contraseñas y datos de usuario.

* 1. Envío de formularios usando AJAX

Es posible optar con AJAX como mediador del proceso de envío de datos al servidor.



Para tener una diferenciación del Front-End con el Back-End

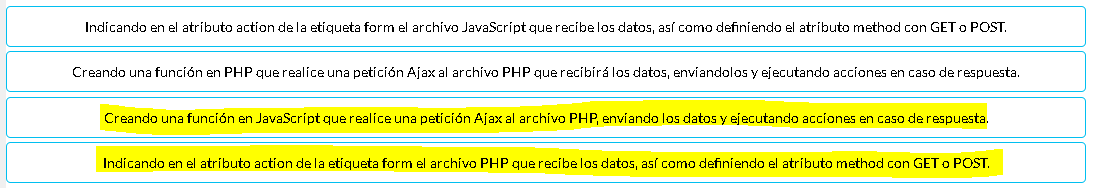


Esto nos permite tener archivos JS, HTML y PHP, en caso de utilizar AJAX, si esto no se hace el Back-end se vería forzado a trabajar con archivos PHP solamente. Para organización y mantenibilidad del código

* 1. Actividad

Tiempo de pensar

En esta lección hemos hablado sobre los formularios y la forma en que se envían los datos a un archivo PHP. Uno de los conceptos más importantes es la diferenciación entre los métodos de envío de datos GET y POST, donde el primero enviaba los datos en la URL del archivo de destino, y la segunda los enviaba en el cuerpo de la petición sin mostrarlos en la URL.  
  
Instrucciones:  
  
Selecciona dos respuestas válidas para la pregunta que se muestra a continuación:  
  
Pregunta:  
  
¿Cómo se envían los datos desde una página HTML a un archivo PHP?



* 1. Ejercicio de Codificación

HORA DE codificar

Los formularios son la herramienta más importante para capturar datos de los usuarios que acceden a un portal web. Sin embargo, es necesario saber qué hacer con dichos formularios para que la información diligenciada pueda ser almacenada en algún lugar, o ya bien usada para la interacción del usuario con el sistema.  
  
En PHP existen dos maneras de trabajar con formularios: directamente entre archivos PHP en los que se mezcla dicho lenguaje con HTML o usando Ajax como intermediario en el proceso de envío y recepción de datos. Aunque ambas posibilidades funcionan correctamente, es recomendable aplicar la segunda en tus proyectos, ya que así puedes diferenciar cada archivo según su funcionalidad, habilitando un mantenimiento más sencillo y limpio del código, así como de darle seguridad al sistema.  
  
En este ejercicio, te encargarás de otorgarle funcionalidad a los formularios de un portal de empleo, en el que se capturan los datos personales, hoja de vida y descripción del profesional. Adicionalmente hay una ventana para ingresar al sistema que también cuenta con un formulario de ingreso.

Contenido

Para el desarrollo de este ejercicio debes descargar el código base llamado codBase.zip. En este paquete encontrarás los archivos HTML, CSS y JavaScript del sistema con funcionalidades propias a la vista. Puedes hacer los cambios necesarios a estos archivos para el desarrollo del ejercicio.

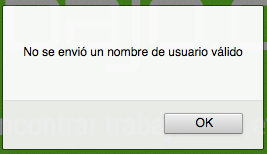
[Descargar archivo base](https://s3.amazonaws.com/nextu-content-production/Desarrollador_Web/06_BackEnd/Ejercicios/WEB16S_C6U3L4_Ejercicio1/solucion/codBase.zip)

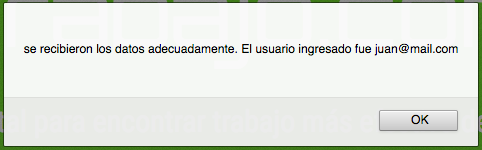
Instrucciones

Crea el archivo login.php que gestionará los datos del formulario login.html.

Crea una función en el archivo login.js para que al enviar el formulario de inicio de sesión, los datos diligenciados se envíen mediante Ajax al archivo login.php. Mostrar en una ventana emergente la respuesta del Back-end.

Verifica, en el archivo login.php, si el dato “usuario” fue diligenciado o no y envia un mensaje a la vista según corresponda.

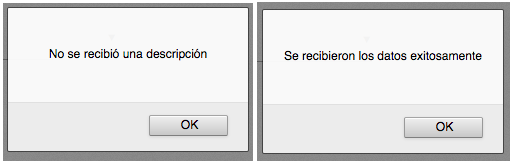




Crea el archivo descripción.php que gestionará los datos del formulario descripcion.html.

Crea una función en el archivo descripcion.js para que al diligenciar el formulario de descripción, los datos se envíen mediante Ajax al archivo descripcion.php. Muestra en una ventana emergente la respuesta del Back-end.

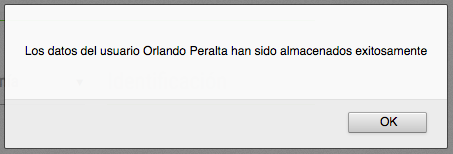
Verifica, en el archivo descripcion.php, si no se enviaron datos del campo descripción y enviar un mensaje como respuesta indicando el resultado de esta verificación a la vista.



Crea el archivo info\_basica.php que gestionará los datos del formulario info\_basica.html. Debes incluir el archivo c\_Usuario.php que contiene la definición de la clas Usuario.

Crea una función en el archivo info\_basica.js para que al enviar el formulario de información básica, los datos diligenciados se envíen mediante Ajax al archivo info\_basica.php. Muestra en una ventana emergente la respuesta del Back-end.

Crea una instancia de la clase Usuario en el archivo info\_basica.php, incluyendo en el constructor todos los datos correspondientes recibidos del formulario. Luego envia el mensaje de respuesta: “Los datos del usuario X Y han sido almacenados exitosamente” en donde X es el atributo nombre del objeto Usuario recién instanciado y Y el apellido.

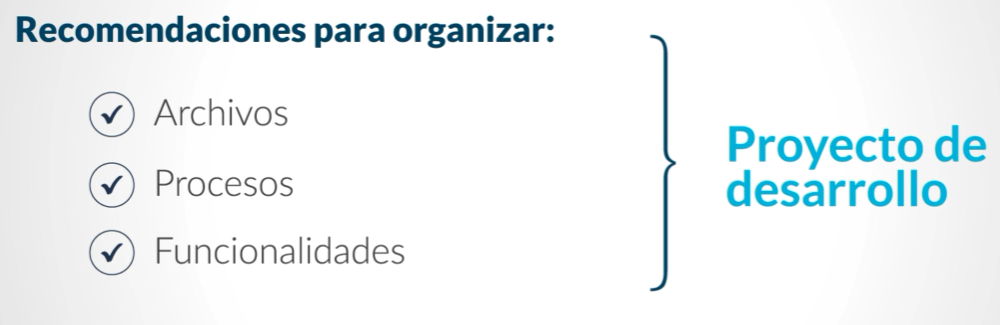


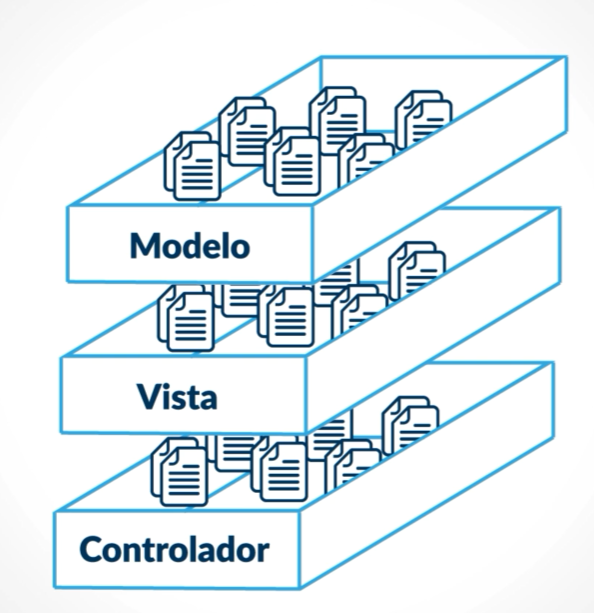
Buenas Prácticas

Diferenciar adecuadamente los archivos de tu proyecto según su funcionalidad, es una de las prácticas más utilizadas en el desarrollo de software, debido a que permite identificar claramente la función de cada elemento en el sistema, aportando a la mantenibilidad y escalabilidad del código.

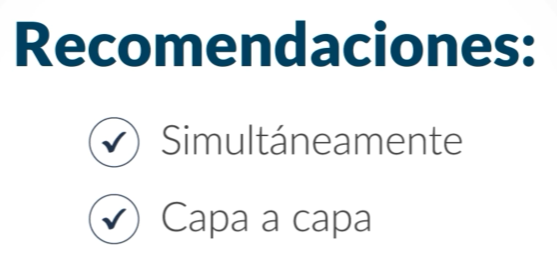
1. Lección 5: Conceptos Avanzados
   1. Patrón MVC – Modelo Vista Controlador

Patrón de Diseño:





* Vista – Interfaz gráfica de usuario: HTML, CSS, JS
* Modelo – Se encarga del procesamiento de los datos, almacenamiento y operaciones que dan funcionalidad al sistema (cerebro), php o backend
* Controlador – Se encarga de comunicar el Modelo con la Vista de un lado a otro. JS con AJAX, peticiones HTTP, etc



* 1. Gestión de Archivos

Operaciones de archivos entre muchas más:



Crear y abrir archivos:

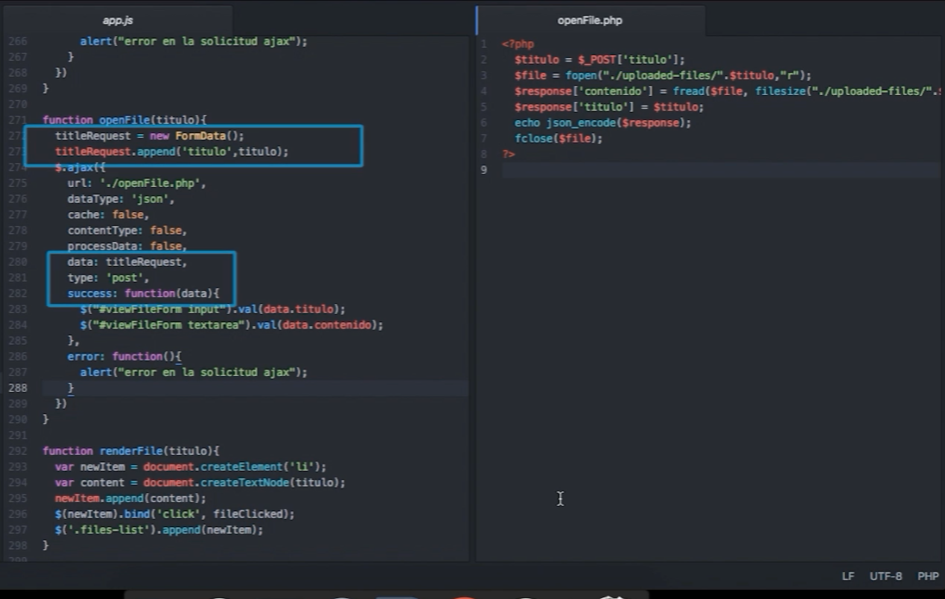
Fopen(‘ruta\_del\_archivo/nombre\_del\_archivo’,’w’) 🡪 (archivo, modo)

Revisar Contenido de un archivo:

Fread(‘$archivoAbierto’,bytes)

Cerrar un archivo:

Fclose($archivoAbierto)



Escribir en un archivo abierto:

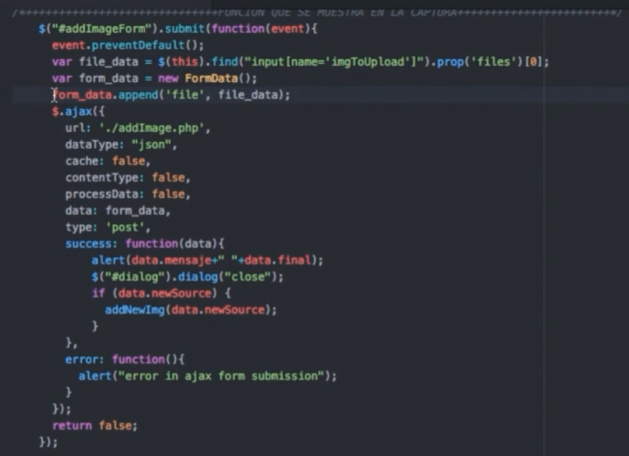
Fwrite($archivoAbierto)

Fwrite($archivoAbierto,$contenidoEscribir)

* 1. Carga de Archivos

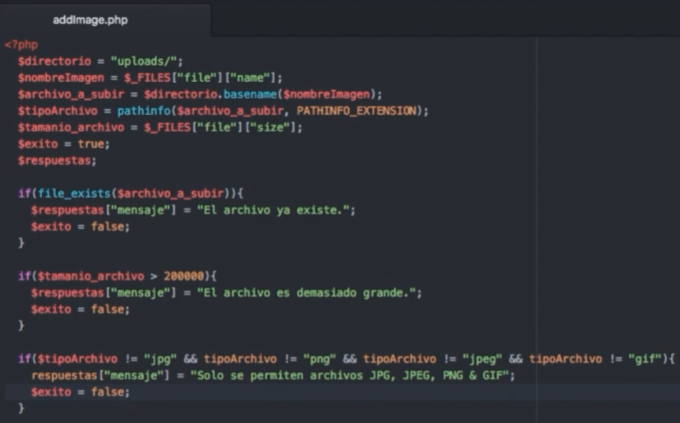
Archivos – utiliza unas características:

Se requiere un Campo input del tipo file



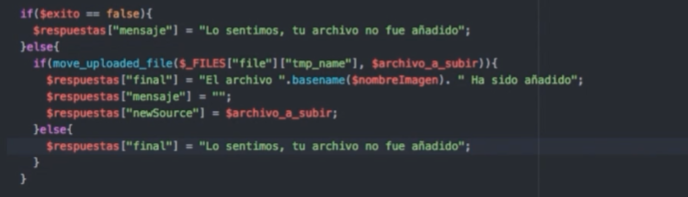
$\_FILES[] 🡪 Almacena los datos provenientes en el AJAX, solo que solamente para archivos

Basename() 🡪 función que obtiene el nombre base en cadena de caracteres.



Para mover un archivo subido a un directorio temporal se utiliza la función move\_uploaded\_file()

Ej: move\_uploaded\_file($FILES[“file”][“tmp\_name”],$archivo\_a\_subir)



En el AJAX

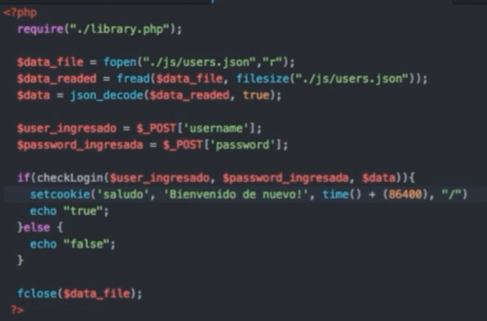


* 1. Variables de Sesión y Cookies

Cookies 🡪 Variables que almacenan valores en el lado del cliente.

Setcookie(“nombre”,”valor”,”tiempo”,”directorio\_donde\_sera\_almacenada”)

Ej:



Isset() 🡪 Función para saber si una variable fue almacenada:

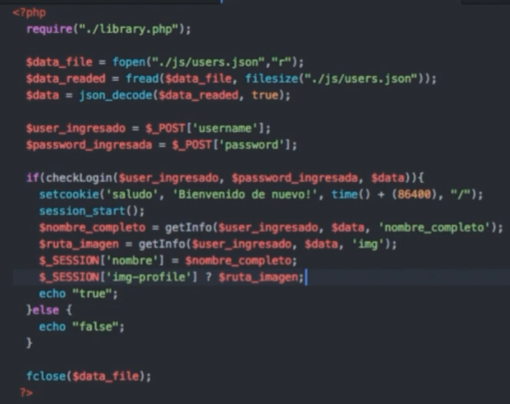
Variables de sesión 🡪 Se almacenan del lado del servidor

Funciones:

Session\_start() 🡪 indica que el presente script esta dentro de la sesión iniciada

$\_SESSION[‘nombre’] 🡪 Para modificarla, borrarla o acceder a ella

Session\_destroy() 🡪 Para indicar al server que dichos datos pueden ser eliminados.



* 1. Actividad 1

Tiempo de pensar

En los videos de esta lección pudiste aprender todo lo que PHP puede realizar con los archivos y las funciones que le permiten abrir, leer, escribir y subir archivos, otorgando a los sistemas web que desarrolles, capacidades muy avanzadas para ofrecer a los usuarios.  
  
Instrucciones:  
  
Observa el siguiente código y lee la pregunta que se hace sobre él. Luego elige dos respuestas para dicha pregunta.  
  
Código: (lenguaje: php)

#### PHP

**<?php**

**$archivo1 = fopen('files/archivo1.txt', 'r');**

**fwrite($archivo1, 'Hola, soy un texto.');**

**?>**

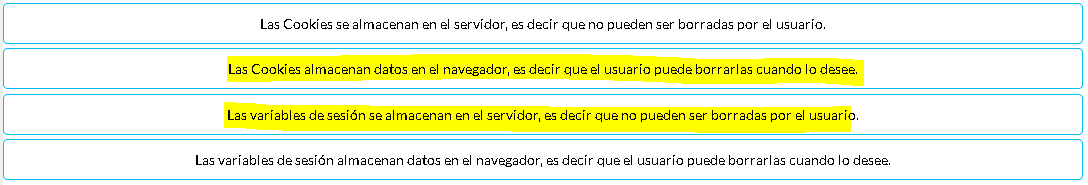
#### Pregunta: Si se desea añadir un texto, sin sobreescribir su contenido, al documento archivo1.txt ¿Qué errores en el código anterior impiden hacerlo correctamente?

#### 

* 1. Actividad 2

#### Tiempo de pensar

#### En este video aprendiste qué son las Cookies y las variables de sesión y cómo ayudan en gran medida al almacenamiento de información persistente entre diferentes archivos PHP de una misma sesión. Las cookies se almacenan en el arreglo asociativo $\_COOOKIE y las variables de sesión en el arreglo asociativo $\_SESSION. Instrucciones: Lee la pregunta que se hace relacionada con las Cookies y las variables de sesión y luego elige dos respuestas incorrectas a la pregunta. Pregunta: Si ambas almacenan datos, ¿Cuál es la diferencia entre una cookie y una variable de sesión?



* 1. Ejercicio de codificación

#### HORA DE codificar

#### 60 MINUTOS

#### La mayoría de los sitios web en la actualidad, basan su funcionalidad de acuerdo con el tipo de usuarios que acceden a las plataformas, otorgando un nivel personalizado de la información y seguridad en el acceso a los datos. El manejo de sesiones y cookies amplía la cantidad de funcionalidades que se le pueden añadir a un portal web, haciendo posible el manejo de usuarios y el transporte de datos entre diferentes archivos PHP en una misma sesión. Por otro lado, la gestión de archivos que permite a un usuario subir ficheros al servidor, tales como imágenes, documentos, entre otros, añade a los sistemas web la capacidad de crear un canal de comunicación de archivos entre el cliente y el servidor. Tanto el manejo de sesiones, cookies, y archivos se realiza en PHP mediante arreglos asociativos, respectivamente: $\_SESSION, $\_COOKIE, $\_FILES. En este ejercicio, vas a completar el portal de empleo iniciado en el ejercicio de la lección anterior. En esta ocasión debes añadir la funcionalidad de abrir, leer y sobreescribir un archivo que almacena los usuarios del sistema, así como usar las sesiones para controlar su ingreso; y permitir la subida de archivos al servidor. Recuerda, para el trabajo de lectura y escritura de archivos, darle permisos, a todos los usuarios, de lectura y escritura a los archivos y carpetas en el servidor, relacionadas con este proceso.

#### Contenido

#### Para el desarrollo de este ejercicio debes continuar con el ejercicio anterior, por lo que tu punto de partida será el código con el que solucionaste el ejercicio de la lección anterior.

#### Instrucciones

#### Crea el archivo library.php en el cual incluirás funciones útiles que puedas utilizar en el proyecto. En él, crea dos funciones: La primera debe abrir el archivo users.json en la carpeta data, leerlo y retornar sus datos como un arreglo asociativo. Para convertir el archivo leído en un arreglo asociativo, debes usar la función json\_decode indicando en su primer parámetro el archivo leído y en el segundo true. La segunda función debe recibir tres parámetros: un nombre de usuario (username), una contraseña y un arreglo asociativo con los datos del archivo users.json. La función debe verificar que el usuario ingresado como parámetro sea igual a un usuario almacenado en el arreglo asociativo y posteriormente, que la contraseña ingresada como parámetro corresponda a la contraseña de ese usuario en el arreglo asociativo.

#### Remueve el mensaje de éxito en el archivo login.php y, en su defecto, invocar la función de verificación creada en el paso anterior. En caso de haber coincidencia, inicia una sesión, crea una variable de sesión con el nombre de usuario (username) de la persona que ha ingresado y redirige a la página “index.html”. En caso de no coincidir el usuario y la contraseña, se debe mostrar un mensaje de error en una ventana emergente.

#### Crea una función en la librería, que a partir de un nombre de usuario (username) y un campo, retorne el valor de ese campo para el usuario indicado del archivo users.json.

#### Crea un archivo llamado “checkSession.php” en el que verifiques si la variable de sesión creada en el paso anterior ha sido definida. Si esto es verdadero, se debe retornar a la vista, usando la función de la librería creada en el paso anterior, los siguientes campos: nombre, descripcion, id, hoja\_vida y profile\_img. Si es falso, se debe hacer una redirección a la página de login.html debido a que no se ha identificado una sesión iniciada.

#### Crea en el archivo index.js una función que reciba como parámetros los datos recibidos del modelo en caso de detectarse una sesión iniciada. El título de la página y el ícono en el extremo derecho en la barra superior adquieren el valor del nombre que se encuentre en el archivo users.json. Si se detecta una descripción, un id y una hoja de vida en el archivo users.json se debe mostrar un ícono que indique que ya se ha diligenciado, en caso contrario debe mostrarse una equis (x). La sección Perfil Profesional contiene la descripción del usuario en el archivo users.json, en caso de no haber una, se muestra un mensaje. Si alguno o ninguno de los datos se ha definido en el archivo user.json para el usuario que ha ingresado, la página index.html debe verse así:

#### 

#### Si por el contrario sí hay datos para los campos mencionados se debe ver la página de inicio así:

#### 

#### Crea una o más funciones en la librería que a partir de un nombre de usuario (username), el nombre de un campo y un valor para ese campo, sobrescriban el archivo users.json y actualicen el campo especificado con el valor dado en los parámetros para el usuario especificado.

#### Remueve el mensaje de éxito del archivo descripcion.php y en esta parte actualiza, usando la función creada en el paso anterior, los campos categoría y descripción con los valores recibidos del formulario para el nombre de usuario (username) almacenado en la variable de sesión.

#### Incluye el código necesario para enviar en la petición Ajax del archivo info\_basica.js, la imagen que se incluye en el formulario. Adicionalmente crea en la librería una función que reciba como parámetro la llave con la que se envió la imagen desde la vista en el arreglo asociativo $\_FILES, y la carpeta en el servidor en la que se almacenará dicha imagen; y verifica si ya existe el archivo en el servidor, si tiene un tamaño inferior a 10 MB y si contiene un formato de tipo imagen o documento de texto (jpg, jpeg, png, doc, txt, docx, pdf), antes de subirla a la carpeta indicada. Se debe retornar un arreglo asociativo con los posibles mensajes de error y de éxito, así como la ruta del archivo recién subido.

#### Invoca la función creada en el paso anterior para subir la imagen proveniente del formulario info\_basica.html. Adicionalmente sobreecribe el archivo users.json con todos los datos almacenados en el objeto de la clase Usuario ya instanciado desde el ejercicio anterior.

#### Crea una función que realice una petición Ajax en el archivo hoja\_vida.js y envíe el documento ingresado. En el modelo, usa la función de la librería para subir archivos, con el fin de subir el archivo proveniente del formulario en la vista. Los documentos subidos como hoja de vida deben almacenarse en la carpeta uploadedDocs.

#### Usa la función creada en el paso 5 para todas las demás páginas del portal. Realiza la petición ajax al mismo archivo checkSession.php y modifica la vista de acuerdo a como lo necesite cada página, al final tendrás algo así:

#### 

#### 

#### 

#### 

#### 

#### Crea el archivo logout.php en el que se cierre la sesión se redirija a la página login.html

#### Buenas Prácticas

#### Cuando te enfrentas a proyectos un poco más grandes y robustos, es muy posible que te encuentres con archivos PHP con gran cantidad de código. Es muy recomendable que hagas uso de las funciones, objetos y la inclusión de archivos, para que dividas el código en diferentes módulos, permitiendo una mejor organización del proyecto y reusabilidad de funcionalidades.

1. UNIDAD 3: Prueba