PRACTICAS PHP 8 AVANZADO

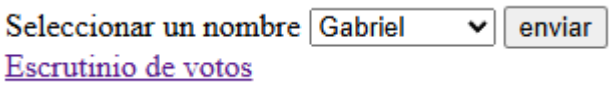
1) Pruebas con las funciones de **arrays**.
    a. Array_map, filter, etc.
    b. Utilizar funciones anónimas o flecha.

2) Definir variables globales, funciones con parámetros por copia / referencia, acceso a variables globales con global o $GLOBALS, definición de variables locales en funciones con static y sin static. Funciones con parámetros opcionales.

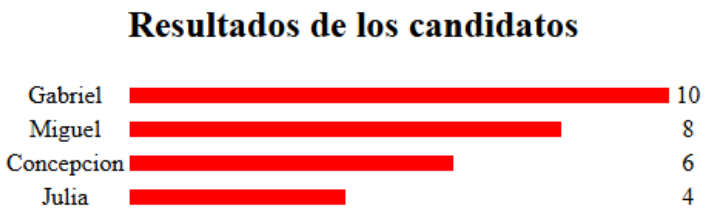
3) Generar etiquetas dinámicas: combos con el contenido de un fichero de meses y una tabla con un fichero CSV. Utilizar un php distinto y probar desde otro php. **require_one 'lib.php'.**

4) Cargar todas las imágenes en un array y publicarlas en una página de PHP. Se puede implementar una función, como **parámetros se enviará el directorio y el número de columnas que tendrá la galería**, será parametrizable.

5) **Ficheros_votos**: A partir del formulario, se va generando un fichero de votos.



    a. Los resultados se mostrarán en el fichero escrutinio.php

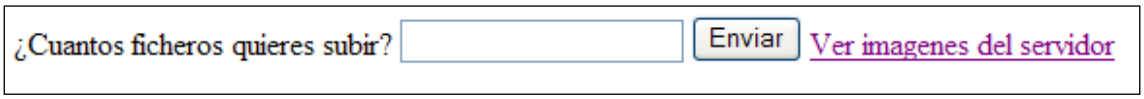


    b. En el fichero **grabar.php** se recoge del formulario el candidato votado y se guarda en el fichero. Abrir el fichero con **fopen** y el **modo a+t**

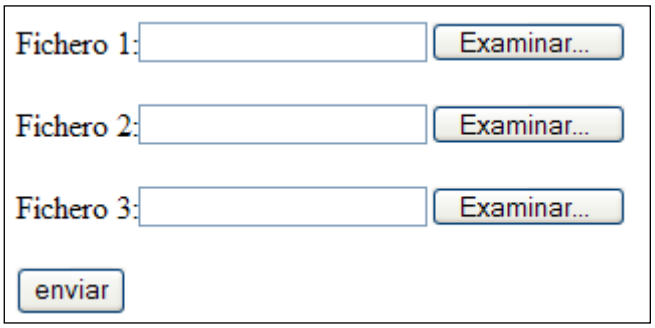
6) Ficheros y directorios:

Mostrar un campo de texto y un botón, el usuario teclea un número se le ofrece la posibilidad de subir tantos ficheros como número haya indicado.

Habrá otro enlace para publicar todas las imágenes del directorio del servidor. Para localizar las imágenes podemos utilizar strpos().



Si el usuario teclea 3:



Al subir los ficheros se puede mostrar datos como el nombre, tipo y tamaño. Utilizar el array $_FILES con los nombres de los input file y los campos: type, name, size, error.

7) **POO**. **Relación de composición** entre las categorías y los productos de la Base de datos. Crear una carpeta para el proyecto que luego se irá extendiendo para llegar a los DAOs con PDO y servicios REST con Slim.
   a. **proyecto_empresa/web/backend/src/data**
   b. La carpeta data (se puede llamar models), en java se llama beans
   c. Contendrá dos ficheros: **Categoria**.php y **Producto**.php con ambas clases.
   d. Implementar constructor, get/set y __toString de las dos clases.
   e. Crear la estructura del proyecto y utilizar composer.

8) **POO**. Relación de **herencia** con clases **abstractas**. Una empresa quiere llevar el control de todas las personas que entran y salen. Por un lado, sin hacer ninguna distinción quiere almacenar el nombre, apellidos y dni. En un nivel más interno hace dos distinciones: el personal externo que

accede a la empresa (mantenimiento, cafetería, etc.) no están en nómina, pero si hay que dejar una constancia de la empresa a la que pertenecen.

En el caso de los empleados todos tienen sueldo, pero se consideran dos perfiles: ingenieros y jefe de proyecto. El ingeniero dispone de 2 pagas extras y el jefe de proyecto incentivos.

De estos dos perfiles si nos interesa calcular el sueldo final de todos. A parte tiene que existir la posibilidad de imprimir los datos por pantalla de todas las personas que acceden a la empresa. Aparte quieren asignar una numeración secuencial a los ingenieros y otra distinta a los jefes de proyecto.

Se pide implementar toda la jerarquía de clases, con propiedades y métodos, eligiendo acceso, herencia, etc. Constructores, destructores, toString, cálculo del sueldo, y algún método que nos diga el número de ingenieros y jefes de proyecto que hay.

Para hacer la prueba se pueden emitir mensajes por pantalla

9) **Excepciones**: A partir de la clase Persona, implementar una excepción personalizada. La nueva excepción se lanzará cuando se modifique la edad de la persona y se intente indicar un valor atípico, como puede ser una edad negativa. En la carpeta practicas/excepciones se encuentra el fichero Persona.php.

10) **DPO**: modificar el ejemplo del PDF para conectar con la BD de datos empresa y realizar un listado de los productos de una determinada categoría.

11) **DPO**: Implementar un patrón DAO con las operaciones CRUD sobre un producto.