IES Los Albares
Desarrollo de Aplicaciones Web

Tarea práctica

PHP DataFilter - Simulador de BBDD

Módulo:	Desarrollo Web en Entorno Servidor
Curso:	2º DAW
Unidad de Trabajo:	UT02/UT03 – Programación en PHP
Fecha:	Octubre 2025

Curso 2025-26

Introducción

En esta actividad práctica desarrollaremos una aplicación web que simula el funcionamiento de una consulta a una base de datos utilizando PHP. El objetivo es crear un sistema de filtrado dinámico donde los usuarios puedan buscar productos según diferentes criterios, aplicando las estructuras de programación fundamentales del lenguaje.

Contexto y Datos Iniciales

La aplicación trabajará con un catálogo de productos simulado. Para ello, utilizaremos un array multidimensional como fuente de datos (la "base de datos"). Ejemplo:

Datos del catálogo de productos

```php
<?php
$productos = [
    [ "id" => 1, "nombre" => "Laptop Pro", "categoria" => "Electronica",
      "stock" => 15, "precio" => 1200.50 ],
    [ "id" => 2, "nombre" => "Teclado Mecánico", "categoria" => "Accesorios",
      "stock" => 50, "precio" => 80.00 ],
    [ "id" => 3, "nombre" => "Monitor 4K", "categoria" => "Electronica",
      "stock" => 10, "precio" => 450.75 ],
    [ "id" => 4, "nombre" => "Silla Gamer", "categoria" => "Mobiliario",
      "stock" => 25, "precio" => 180.00 ],
    [ "id" => 5, "nombre" => "Mouse Óptico", "categoria" => "Accesorios",
      "stock" => 100, "precio" => 25.20 ],
    [ "id" => 6, "nombre" => "Webcam HD", "categoria" => "Electronica",
      "stock" => 30, "precio" => 65.00 ]
];
?>
```

Se proporcionará un fichero con un array similar al mostrado arriba, que se utilizará como fuente de datos para tu aplicación.

Requisitos de la Aplicación

Tu tarea consiste en desarrollar una aplicación web en varios ficheros PHP que combinen HTML y código de servidor para crear un sistema de filtrado de productos interactivo. Se sugiere los siguientes componentes:

- Fichero de datos: `catalogo_productos.php` (contiene el array de productos), te será proporcionado.
- Funciones de filtrado: contiene las funciones PHP necesarias para procesar los datos.
- Interfaz de usuario: fichero principal que genera el formulario HTML y muestra los resultados.
- Procesamiento de solicitudes: lógica para manejar las solicitudes del formulario y aplicar los filtros. Puede estar en el mismo fichero que la interfaz o en uno separado.

Componente 1: Funciones de Selección y Filtrado

Contiene las funciones PHP necesarias para procesar los datos del array de productos y aplicar los filtros seleccionados por el usuario.

</> Funciones a implementar:

- Filtrado numérico:
  - `filtrar_igual($productos, $campo, $valor)`
  - `filtrar_mayor_que($productos, $campo, $valor)`
  - `filtrar_menor_que($productos, $campo, $valor)`
- Filtrado de texto:
  - `filtrar_igual($productos, $campo, $valor)`
  - `filtrar_contiene($productos, $campo, $valor)`
- Selección de campos: `seleccionar_campos($productos, $campos)`. Debe considerar lo siguiente:
  - Si no se especifican campos, un array vacío o *, devolver todos los campos.
  - Si se especifican uno o más campos, devolver un array multidimensional **asociativo** con los campos seleccionados.

Te ayudarán las funciones de:

- `array_filter`: https://www.php.net/manual/es/function.array-filter.php
- `array_map`: https://www.php.net/manual/es/function.array-map.php
- `array_intersect_key`: https://www.php.net/manual/es/function.array-intersect-key.php
- `array_flip`: https://www.php.net/manual/es/function.array-flip.php

Componente 2: Formulario de Filtrado Dinámico

Especificaciones del formulario HTML:
Debes crear un formulario HTML (`<form>`) que envíe una petición a otro componente encargado de procesarla usando POST. El formulario debe incluir:

1. **Campo a Seleccionar:** elemento `<input type=''checkbox''>` cuyas opciones se generen **dinámicamente** usando PHP. Las opciones deben corresponder a las claves del array de productos.

   Pista: Utiliza la función `array_keys()`

2. **Campo a Filtrar:** elemento `<select>` con opciones cuyas opciones se generen **dinámicamente** usando PHP. Las opciones deben corresponder a las claves del array de productos.

3. **Desplegable "Operador":** Un elemento `<select>` con opciones fijas:

   - "Igual a" (para comparaciones exactas de texto o numéricas)
   - "Contiene" (para búsquedas de texto parciales)
   - "Mayor que" (para comparaciones numéricas)
   - "Menor que" (para comparaciones numéricas)

4. **Campo de texto "Valor":** Un `<input type=''text''>` donde el usuario escribirá el criterio de búsqueda.

5. **Botón de envío:** Un `<input type=''submit''>` para procesar la consulta.

Extra: utiliza Javascript para mejorar la usabilidad del formulario, mostrando/ocultando operadores según el tipo de campo seleccionado (numérico o texto).

Componente 3: Lógica de Procesamiento PHP

Procesamiento de solicitudes:
Debe implementar la siguiente lógica:

1. **Detección de envío:** Verificar si el formulario ha sido enviado usando `$_POST`.
2. **Recuperación de datos:** Obtener los cuatro valores del formulario: campo a seleccionar, campo a filtrar, operador y valor.
3. **Selección de función de filtrado:** Según el operador seleccionado, determinar qué función de filtrado utilizar.
4. **Llamar a las funciones:** Ejecutar la función de filtrado seleccionada y luego la función de selección de campos con los resultados obtenidos.
5. **Gestión de la carga inicial:** Si no se ha enviado el formulario, mostrar todos los datos de todos los productos.

Componente 4: Visualización de Resultados

Presentación de datos:
La visualización debe cumplir estos requisitos:

1. **Tabla HTML dinámica:** Crear una `<table>` donde:
   - Los encabezados (`<th>`) se generen automáticamente a partir de las claves del array
   - Las filas (`<tr>`) muestren los datos filtrados

→ Ejemplo de Funcionamiento Esperado

Cases de uso ejemplo

Tu aplicación debe poder realizar consultas como:

- Campo a seleccionar: `nombre, precio`
  Campo a filtrar: `categoria`
  Operador: `Igual a`
  Valor: `Electronica`
  Resultado esperado: Tabla con los nombres y precios de los productos en la categoría "Electronica".
- Campo a seleccionar: `Sin campo`
  Campo a filtrar: `stock`
  Operador: `Mayor que`
  Valor: `20`
  Resultado esperado: Tabla con todos los campos de productos con stock mayor a 20.
- Campo a seleccionar: `nombre`
  Campo a filtrar: `nombre`
  Operador: `Contiene`
  Valor: `Pro`
  Resultado esperado: Tabla con los nombres de productos que contienen "Pro" en su nombre.

→ Ejemplo de Interfaz

id	nombre	categoria	stock	precio
1	Laptop Gamer	Electronica	15	1200.5
2	Teclado Mecánico	Accesorios	50	80
3	Monitor UHD	Electronica	10	450.75
4	Silla Gamer	Mobiliario	25	180
5	Ratón Óptico	Accesorios	100	25.2
6	Webcam UHD	Accesorios	30	65

Campo a seleccionar: ☐ id ☐ nombre ☐ categoria ☐ stock ☐ precio

Campo a filtrar [Sin campo]

Función de filtrado [Sin filtrado]

Criterio [ ]

[Enviar]

Entrega

Formato de entrega

Debes entregar un fichero `.zip` que contenga:

- Un directorio llamado `src/` con todos los ficheros PHP desarrollados.
- Un directorio llamado `screenshots/` con, al menos, las siguientes capturas de pantalla.
  - **Formulario inicial:** 1 captura de los resultados y el formulario antes de aplicar cualquier filtro.
  - **Resultados con filtro aplicado:** 1 captura del filtro seleccionado en el formulario y 1 captura que muestre los resultados después de aplicar el filtro filtro.

Consejos

Recomendaciones

- **Funcionalidad e interfaz básica:** empieza mostrando todos los productos sin filtros y construye el formulario de manera dinámica aunque no tenga funcionalidad.
- **Desarrollo incremental:** comprueba cómo funciona cada parte antes de avanzar. Especialmente la manera en que llegan los datos del formulario en la variable `$_POST`.
- **Asume primero el "happy path"**: implementa la funcionalidad básica antes de añadir validaciones o manejo de errores.

Ayuda

Agrupar valores de checkbox

```
<input type="checkbox" name="campo_a_seleccionar[]" value="nombre_campo_1">
<input type="checkbox" name="campo_a_seleccionar[]" value="nombre_campo_2">
```

Agrupará todos los valores **seleccionados** en un array accesible con la clave "campo_a_seleccionar" . Con valores como

```
Array
(
    [0] => nombre_campo_1
    [1] => nombre_campo_2
)
```

Funciones anónimas

Puedes definir funciones anónimas para pasar a `array_filter()` como en el siguiente ejemplo:

```
$filtrados = array_filter($productos, function($producto) use ($campo, $valor) {
    return $producto[$campo] == $valor;
});
```

A estas funciones se las conoce como *closures* y la palabra clave `use` permite acceder a variables externas al ámbito de la función. Es lo mismo que si pusieras la palabra clave `global` dentro de la función. Una versión más avanzada de este concepto son las funciones flecha (arrow functions). El equivalente en Java serían las expresiones lambda.

```
$filtrados = array_filter($productos, fn($producto) => $producto[$campo] == $valor);
```

Observa que en este caso no hace falta la palabra clave `use` porque las funciones flecha capturan automáticamente las variables del ámbito padre.