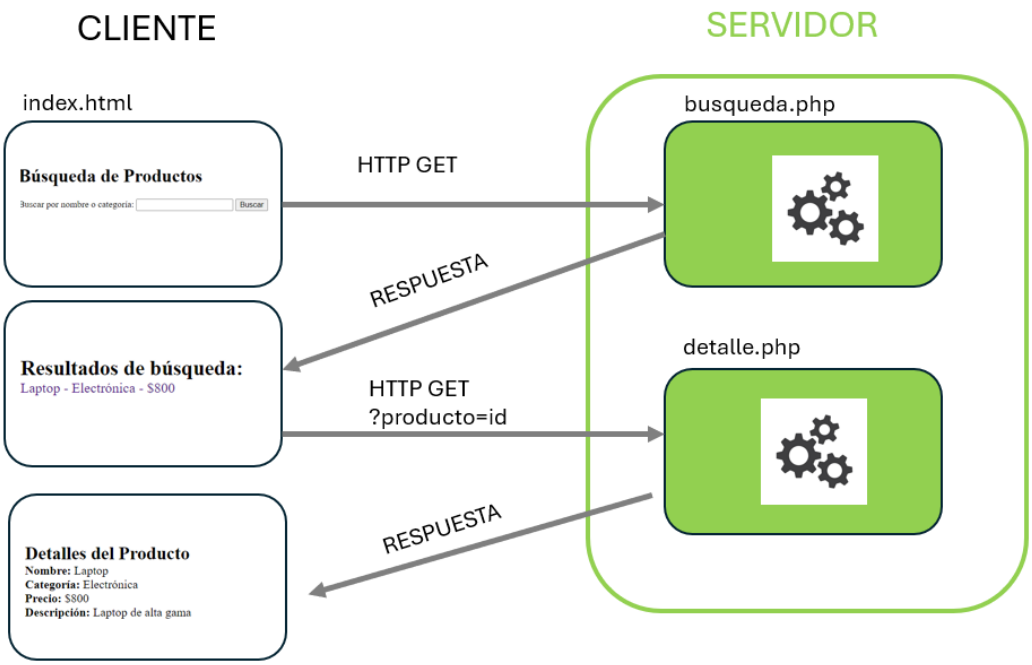
Actividad Desarrollo en Servidor

Proceso de búsqueda y vista de detalles de ítems sitio web

En esta actividad vamos a desarrollar un proceso muy común en la mayoría de las aplicaciones web, como es el buscar un ítem , mostrarlo y luego ver su detalles mediante un enlace.

Fíjate en el siguiente flujo:



En este flujo de interacción **puedes ver la dinámica de búsqueda** de un elemento en general de manera sencilla.

Objetivos de la práctica:

- **Implementar un proceso común en las aplicaciones y sitios web** como es el de búsqueda de ítems por algún criterio y su posterior vista de detalle
- **Comprender flujos de interacción entre módulos php**
- Uso función strpos para buscar ocurrencias de cadenas en campos de texto
- Uso de parámetros en solicitudes GET

- Creación de enlaces <a> dinámicos

Desarrollo de la actividad

A continuación se muestran los módulos php necesarios para implementar esta actividad.

Implementa y desarrolla todos los módulos de la actividad (algunos de ellos pueden tener fragmentos de código que tienes que completar)

Haz una prueba de búsqueda para ver que funciona correctamente y contesta a las cuestiones que se plantean al final.

index.html

Nota: Fíjate en el diagrama de transición de la introducción y completa el formulario enviándoselo al módulo php adecuado.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Búsqueda de Productos</title>
</head>
<body>
    <h1>Búsqueda de Productos</h1>
    <form method="GET" action= <!-----! ">
        <label for="busqueda">Buscar por nombre o categoría:</label>
        <input type="text" name="busqueda" id="busqueda">
        <button type="submit">Buscar</button>
    </form>
```

```
</body>
</html>
```

Módulo datos.php

Este módulo simula una base de datos mediante un array asociativo donde cada entrada es un id de producto y su valor son sus campos descriptivos.

```
<?php
// Simula una base de datos de productos de diferentes categorías
$productos = [
    1 => ["nombre" => "Laptop", "categoría" => "Electrónica", "precio" => 800, "descripción" => "Laptop de alta gama"],
    2 => ["nombre" => "Smartphone", "categoría" => "Electrónica", "precio" => 500, "descripción" => "Teléfono inteligente con buena cámara"],
    3 => ["nombre" => "Cafetera", "categoría" => "Hogar", "precio" => 100, "descripción" => "Cafetera automática con temporizador"],
    4 => ["nombre" => "Libro", "categoría" => "Libros", "precio" => 20, "descripción" => "Libro de ciencia ficción"],
    // Añadir más productos si es necesario
];
```

Módulo busqueda.php

Este módulo obtiene el nombre del producto o categoría y lanza el proceso de una búsqueda en el servidor . Completa las partes de código indicadas con "----------"

NOTA . Recuerda La función isset() en PHP se utiliza para verificar si una variable está definida y no es null. En este caso se usa para saber si se ha pasado un valor de búsqueda desde el formulario

```
<?php
include 'datos.php';
    if (isset($_GET['busqueda'])) {
```

```
        $busqueda = strtolower(--------);

        echo "<h2>Resultados de búsqueda:</h2>";

        echo "<ul>";

        foreach ($productos as $id => $producto) {

            if (strpos(strtolower(--------), $busqueda) !==
false || strpos(strtolower(-------), $busqueda) !== false) {

                echo "<li><a href='detalles.php?producto=$id'>
Ver detalles {$producto['nombre']} - {$producto['categoria']}
</a></li>";

            }

        }

        echo "</ul>";

    }
```

Fíjate cómo se busca con la función **strpos** tanto en el campo nombre como el campo categoría

```
if (strpos(strtolower($producto['nombre']), $busqueda) !== false
|| strpos(strtolower($producto['categoria']), $busqueda)
```

Fíjate también cómo el resultado de la búsqueda **arma un enlace <a href> para luego ver los detalles. En este enlace irá toda la información para recuperar los detalles**

```
echo "<li><a href='detalles.php?producto=$id'> Ver detalles
{$producto['nombre']} - {$producto['categoria']} </a></li>";
```

Módulo detalles.php

El módulo detalles simplemente muestra la información del item que se ha solicitado a partir de su id.

```
<?php

include 'datos.php';

if (isset($_GET['producto'])) {

   $idProducto = $_GET['producto'];
```

```
    if (isset($productos[$idProducto])) {
        $producto = $productos[$idProducto];
        echo "<h2>Detalles del Producto</h2>";
        echo "<p><strong>Nombre:</strong> {$producto['nombre']}</p>";
        echo "<p><strong>Categoría:</strong> {$producto['categoria']}</p>";
        echo "<p><strong>Precio:</strong> \${$producto['precio']}</p>";
        echo "<p><strong>Descripción:</strong>
{$producto['descripcion']}</p>";


    } else {
        echo "<p>Producto no encontrado.</p>";
    }
}
```

Cuestiones sobre la actividad:

1. En el módulo de detalle haz que al final se muestre un enlace a index.html para poder volver al inicio

Validación de datos

2. ¿Qué sucede si el campo de búsqueda está vacío o se ingresan caracteres no válidos? ¿Qué ocurre si el campo de búsqueda lleva espacios? **Agrega validación para asegurar que el campo de búsqueda no esté vacío y que se manejen entradas no válidas** de manera adecuada

   NOTA :Usa las funciones **empty() y trim()**

Sensibilidad de mayúsculas y minúsculas

- ¿Cómo se comporta el buscador si se introducen términos en mayúsculas o minúsculas? ¿Es la búsqueda sensible a las mayúsculas?
- **NOTA Usa strtolower() o stripos() para garantizar independizar la busqueda.**

Seguridad

¿Cómo podrías mejorar la seguridad de la búsqueda contra ataques como la inyección de código o entradas maliciosas?

Implementen buenas prácticas de seguridad, como la sanitización de entradas de usuarios, para evitar vulnerabilidades como ataques de inyección de código.

```
$busqueda = htmlspecialchars(trim($_GET['busqueda']), ENT_QUOTES, 'UTF-8');
```

- La función htmlspecialchars() en PHP se utiliza como medida de seguridad para evitar ataques de **inyección de código** o **XSS (Cross-Site Scripting)**, protegiendo así la integridad de la aplicación web y los usuarios.

Ejemplo de la salida de la función <script>alert('Hackeado!');</script>

```
&lt;script&gt;alert(&#039;Hackeado!&#039;);&lt;/script&gt;
```

NOTA la Sanitización consiste en limpiar o transformar los datos para hacerlos seguros. El objetivo es eliminar o transformar los caracteres que puedan representar un riesgo.

Convertir caracteres especiales de HTML (<, >, &, etc.) en entidades HTML para evitar la inyección de código.

Como vimos antes con htmlspecialchars(), esta técnica escapa caracteres como <, >, ", y ' para evitar que sean interpretados como código HTML o JavaScript.

Remover caracteres peligrosos:

- Usar funciones como filter_var() para eliminar caracteres no deseados de entradas como correos electrónicos, URLs, o números de teléfono
- Ejemplos para sanitizar el correo electrónico y un campo

```
$email = filter_var($emailInput, FILTER_SANITIZE_EMAIL);

// Fozar a que un campo sea numérico

$numero = filter_var($input, FILTER_SANITIZE_NUMBER_INT);
```

La sanitización de la entrada de usuario es crucial para evitar vulnerabilidades que podrían comprometer la seguridad de tu aplicación. Asegurarse de que los datos ingresados no puedan ser usados para inyectar código malicioso o causar errores es una práctica fundamental para cualquier desarrollo web seguro.