IES Luis Vélez de Guevara

Dpto. de Informática

Ferretería

Antonio Aguilar Humanes

Horas de libre configuración

D.ª María de los Ángeles Rider Jiménez

2º ASIR I.E.S LUIS VÉLEZ DE GUEVARA

Curso 2024/2025

Índice

1. Descripción

Descripción General

La web es una aplicación para una ferretería que combina un diseño atractivo, funcionalidad dinámica y seguridad en la gestión de usuarios. Su propósito principal es facilitar la visualización y gestión de productos, así como ofrecer a los clientes una experiencia de compra intuitiva y ordenada.

Características Clave

1. *Gestión de Usuarios y Autenticación:*
   - **Inicio de Sesión**: Los usuarios pueden iniciar sesión en la plataforma utilizando un sistema seguro basado en sesiones de PHP. Esto garantiza que solo los usuarios autorizados puedan acceder a las secciones protegidas de la web.
   - **Opciones para Usuarios Nuevos**: Una opción para los nuevos usuarios les permite registrarse fácilmente en la plataforma mediante un formulario en una página de registro.
2. *Roles de Usuario:*
   - Los usuarios están clasificados en roles (por ejemplo, cliente o administrador).
   - Los clientes pueden acceder a funciones específicas, como añadir productos a su cesta de compras.
   - Los administradores pueden gestionar la base de datos de productos (editar, añadir o eliminar).
3. *Gestión de Productos:*
   - Los productos están organizados en una base de datos MySQL.
   - Se pueden mostrar todos los productos disponibles en una tabla con detalles como el nombre, descripción y precio.
   - Los clientes tienen la opción de añadir productos a una cesta de compras, facilitando la experiencia de compra.
4. *Base de Datos y Conexión:*
   - La base de datos se conecta mediante un archivo `db.php` que almacena las credenciales de conexión de manera eficiente y centralizada.
   - Las consultas a la base de datos permiten listar productos, validar usuarios y gestionar compras.
5. *Diseño y Usabilidad:*
   - Utiliza **Bootstrap** para hacer el diseño atractivo y responsive, asegurando que la web sea compatible con dispositivos móviles y pantallas de diferentes tamaños.
   - Incluye elementos visuales como una imagen de fondo semitransparente y botones estilizados para una mejor experiencia de usuario.
6. *Protección y Seguridad:*
   - Los usuarios no autenticados son redirigidos automáticamente al formulario de login.
   - Se utiliza una combinación de sesiones y cookies para manejar la autenticación y recordar a los usuarios que eligen permanecer conectados.
   - La lógica protege páginas importantes para que sean accesibles únicamente por usuarios con el rol correspondiente.
7. *Navegación Intuitiva:*
   - Página de inicio con mensaje de bienvenida e instrucciones claras.
   - Botones de navegación simples para llevar a los usuarios a las secciones más importantes, como el formulario de login o la lista de productos.

Ejemplo de Flujo del Usuario

1. Un nuevo cliente llega a la página de inicio (`index.php`) y ve la frase "Bienvenidos a la

Ferretería" acompañada de la frase "Tu ferretería de confianza" y un botón para iniciar sesión.
2. Si es un usuario nuevo, puede hacer clic en el botón "Registrarse" desde la página de login para crear su cuenta.
3. Un cliente registrado inicia sesión, es redirigido a la página donde puede explorar los productos disponibles y añadirlos a su cesta de compras.
4. Un administrador puede iniciar sesión y acceder a un menú especial para gestionar productos en la base de datos.

Aspecto Técnico

- Lenguaje y Herramientas: PHP (procedimental), MySQL, HTML, CSS (Bootstrap), JavaScript.
- **Estructura Modular**: Los scripts están organizados para facilitar la reutilización y el mantenimiento. Por ejemplo, el archivo `db.php` centraliza la conexión a la base de datos y puede incluirse en cualquier página.
- **Compatibilidad**: Diseñada para funcionar en entornos locales como XAMPP o WAMP y fácilmente adaptable a servidores en la nube.

Ventajas de la Web

- **Eficiencia**: Proporciona acceso rápido a los datos y permite operaciones dinámicas como la adición de productos a la cesta.
- **Escalabilidad**: La estructura de la web permite agregar funciones adicionales, como un sistema de pago o estadísticas para los administradores.
- **Seguridad**: El uso de sesiones, cookies y la validación de usuarios garantizan que solo las personas autorizadas accedan a las áreas sensibles.

Análisis y Explicación del Código

2. Admin_menu.php

Propósito General del Script

El objetivo de este archivo es permitir a los usuarios autenticados (clientes) añadir productos a su cesta desde la base de datos. Está diseñado para ser seguro (validación de sesión) y funcional (gestión de la cesta en tiempo real con sesiones).

Inicio de Sesión y Verificación de Administrador

```php
<?php
// Iniciar la sesión
session_start();

// Verificar si el usuario es administrador
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'administrador') {
    header("Location: login.php");
    exit();
}
?>
```

session_start()

`session_start()`: Inicia una sesión en PHP para que se pueda acceder a las variables de sesión, como `$_SESSION['usuario']` y `$_SESSION['rol']`.

Verificación de Administrador:

- Comprueba si el usuario ha iniciado sesión (`isset($_SESSION['usuario'])`) y si su rol es "administrador" (`$_SESSION['rol'] == 'administrador'`).
- Si no cumple estas condiciones, el usuario es redirigido a `login.php` mediante `header("Location: login.php");` y el script se detiene con `exit();`.
- **Propósito**: Asegurar que solo los usuarios con rol "administrador" tengan acceso a esta página.

Estructura HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Menú del Administrador</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;
        }
    </style>
</head>
```

***metaetiquetas HTML**:*

- `lang="es"`: Define el idioma de la página como español.
- `meta charset="UTF-8"`: Especifica la codificación de caracteres como UTF-8 para soportar acentos y caracteres especiales.

***Bootstrap CSS**:*

- Incluye la biblioteca CSS de Bootstrap para un diseño responsivo y estilizado.

***Estilos Personalizados**:*

- `padding-top: 50px`: Agrega un espacio superior al contenido del cuerpo para mejorar el diseño.

Contenido del cuerpo

```
<body>
    <div class="container">
        <h1 class="text-center mt-5">Menú del Administrador</h1>
        <div class="list-group mt-3">
            <!-- Enlaces a las diferentes funcionalidades del administrador -->
            <a href="modificar_datos.php" class="list-group-item list-group-item-action">Modificar Producto</a>
            <a href="modificar_datos.php" class="list-group-item list-group-item-action">Eliminar Producto</a>
            <a href="validar.php" class="list-group-item list-group-item-action">Validar Administradores</a>
            <a href="logout.php" class="list-group-item list-group-item-action">Cerrar Sesión</a>
        </div>
    </div>
```

***Estructura de Contenedor**:*

- Usa una clase `container` para centrar y alinear el contenido dentro de una caja responsiva proporcionada por Bootstrap.
- Un encabezado (`<h1>`) saluda al administrador con "Menú del Administrador" y está centrado en la página (`text-center`).

***Lista de Funcionalidades del Administrador**:*

- Utiliza una lista (`list-group`) para mostrar las opciones del menú, con clases de Bootstrap:
  - `list-group-item`: Da estilo a los elementos de la lista.
  - `list-group-item-action`: Hace que los enlaces sean clicables.
- **Opciones Disponibles**:
  - **Modificar Producto**: Dirige a `modificar_datos.php` para editar productos existentes.
  - **Eliminar Producto**: También dirige a `modificar_datos.php`, aunque esto puede ser un error en el enlace ya que "Eliminar Producto" probablemente debería tener su propio script.
  - **Validar Administradores**: Dirige a `validar.php`, una página donde se validan las solicitudes de nuevos administradores.
  - **Cerrar Sesión**: Dirige a `logout.php` para terminar la sesión actual.

Scripts para Interactividad

```
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

3. Añadir_cesta.php

Descripción del código

Este archivo sirve para manejar la funcionalidad de añadir productos a la cesta de compras en una

aplicación web diseñada para clientes autenticados.

Inicio de Sesión y Autenticación

```php
<?php
session_start();
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'cliente') {
    header("Location: login.php");
    exit();
}
```

Session_start()
`session_start()`:

- Inicia la sesión para acceder a las variables de sesión (`$_SESSION`).

Verificación del usuario:

- Comprueba si el usuario está autenticado (`isset($_SESSION['usuario'])`).
- Verifica que el rol del usuario sea "cliente" (`$_SESSION['rol'] != 'cliente'`).
- Si no se cumplen estas condiciones, redirige al usuario a `login.php` y detiene la ejecución del script con `exit();`.
- **Propósito**: Esto protege la página y asegura que solo los clientes tengan acceso a esta funcionalidad.

Manejo de la Petición y Lógica Principal

```php
if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['id'])) {
    include('db.php');
    $producto_id = $_GET['id'];  // ID del producto seleccionado
```

Petición GET:

- El script verifica que se haya hecho una solicitud HTTP de tipo `GET`.
- Obtiene el ID del producto enviado a través del parámetro `id` en la URL (`$_GET['id']`).

`include('db.php')`:

- Incluye el archivo que maneja la conexión a la base de datos.

Consulta del Producto en la Base de Datos

```php
// Consultar el producto seleccionado en la base de datos
$sql = "SELECT * FROM productos WHERE id=$producto_id";
$result = mysqli_query($conn, $sql);
$producto = mysqli_fetch_assoc($result);
```

Consulta SQL:

- Selecciona el producto con el ID proporcionado (`$producto_id`) desde la tabla `productos`.
- El resultado se almacena en `$result` y se convierte en un arreglo asociativo con `$result->fetch_assoc()`.

Verificación del Producto

```php
// Verificar si el producto fue encontrado
if ($producto) {
    $nombre = $producto['nombre'];
    $precio = $producto['precio'];
    $cantidad = 1;  // Cantidad predeterminada
```

Si el producto fue encontrado en la base de datos:

- Extrae su nombre (`$producto['nombre']`) y precio (`$producto['precio']`).
- Define una cantidad predeterminada de "1" para el producto.

Gestión de la Cesta

```php
// Obtener la cesta del cliente desde la sesión o crear una nueva
$cesta = isset($_SESSION['cesta']) ? $_SESSION['cesta'] : [];
if (isset($cesta[$producto_id])) {
    $cesta[$producto_id]['cantidad'] += $cantidad;  // Aumentar cantidad si ya está en la cesta
} else {
    $cesta[$producto_id] = [
        'id' => $producto_id,
        'nombre' => $nombre,
        'precio' => $precio,
        'cantidad' => $cantidad,
    ];
}

$_SESSION['cesta'] = $cesta;  // Actualizar la cesta en la sesión
```

Cesta en la Sesión:

- Si la variable de sesión `$_SESSION['cesta']` existe, la asigna a `$cesta`. Si no, crea un arreglo vacío.
- Si el producto ya está en la cesta (`isset($cesta[$producto_id])`), incrementa la cantidad.
- Si el producto no está en la cesta, lo añade con sus detalles (`id`, `nombre`, `precio`, `cantidad`).
- Actualiza la variable de sesión `$_SESSION['cesta']` con la nueva información.

Mensajes de Éxito o Error

```
    echo "<div class='alert alert-success text-center mt-3'>Producto añadido a la cesta.</div>";
} else {
    echo "<div class='alert alert-danger text-center mt-3'>Producto no encontrado.</div>";
}
```

- Si el producto se agregó exitosamente, muestra un mensaje de éxito con clase de Bootstrap `alert-success`.

- Si no se encuentra el producto en la base de datos, muestra un mensaje de error con la clase `alert-danger`.

Cierre de la Conexión

```
mysqli_close($conn);  // Cerrar la conexión con la base de datos
```

Finaliza la conexión con la base de datos para liberar recursos.

Contenido HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Añadir a la Cesta</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;

        }
    </style>
```

Estructura HTML:

- Define el idioma como español (`lang="es"`) y la codificación UTF-8 para caracteres especiales.
- Incluye Bootstrap para proporcionar estilos modernos y responsivos.

Interfaz del Usuario

```
<body>
    <div class="container">
        <h1 class="text-center mt-5">Añadir a la Cesta</h1>
        <div class="text-center mt-3">
            <a href="listado_productos.php" class="btn btn-primary">Volver a la Lista de Productos</a>
            <a href="ver_cesta.php" class="btn btn-success">Ver Cesta</a>
        </div>
    </div>
```

Muestra un encabezado (`<h1>`) con el título "Añadir a la Cesta".
Incluye dos botones:

1. **Volver a la Lista de Productos**: Redirige a `listado_productos.php`.
2. **Ver Cesta**: Redirige a `ver_cesta.php`, donde probablemente se mostrará el contenido actual de la cesta.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

4. Confirmar_cesta.php

Descripción del codigo

Este archivo gestiona el proceso de compra, desde mostrar el contenido de la cesta hasta registrar la venta en la base de datos y actualizar el stock. También asegura una experiencia clara para el cliente, con mensajes y resúmenes de los productos comprados.

Inicio de Sesión y Verificación del Usuario

```php
<?php
session_start();
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'cliente') {
    header("Location: login.php");
    exit();
}
```

`session_start()`:

- Inicia la sesión del usuario para poder acceder a las variables de sesión, como `$_SESSION['usuario']` y `$_SESSION['rol']`.

Verificación del Cliente:

- Comprueba si hay un usuario autenticado (`isset($_SESSION['usuario'])`) y si el rol de este usuario es "cliente".
- Si no se cumplen estas condiciones, redirige al usuario a `login.php` y detiene la ejecución del script (`exit();`).
- **Propósito**: Asegura que solo los clientes puedan acceder a esta página.

Verificar y procesar solicitudes GET

```php
if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['id'])) {
    include('db.php');
    $producto_id = $_GET['id'];  // ID del producto seleccionado
```

Se comprueba si la solicitud recibida es de tipo `GET` y si el parámetro `id` está presente en la URL.

`include('db.php')`: Se incluye el archivo de conexión a la base de datos.

Se obtiene el valor de `id` enviado en la URL y se almacena en `$producto_id`.

Consultar un producto en la base de datos

```php
// Consultar el producto seleccionado en la base de datos
$sql = "SELECT * FROM productos WHERE id=$producto_id";
$result = mysqli_query($conn, $sql);
$producto = mysqli_fetch_assoc($result);
```

- Construye una consulta para buscar el producto con el ID especificado (`$producto_id`).
- `mysqli_query($conn, $sql)`: Ejecuta la consulta en la base de datos.
- `mysqli_fetch_assoc($result)`: Obtiene la fila resultante como un array asociativo y la almacena en `$producto`.

Comprobar si el producto existe

```php
// Verificar si el producto fue encontrado
if ($producto) {
    $nombre = $producto['nombre'];
    $precio = $producto['precio'];
    $cantidad = 1;  // Cantidad predeterminada
```

- Se verifica si `$producto` contiene información (es decir, si el producto existe en la base de datos).
- Se extraen los valores `nombre` y `precio` del producto y se inicializa una cantidad predeterminada de 1.

Gestionar la cesta del cliente

```php
// Obtener la cesta del cliente desde la sesión o crear una nueva
$cesta = isset($_SESSION['cesta']) ? $_SESSION['cesta'] : [];
if (isset($cesta[$producto_id])) {
    $cesta[$producto_id]['cantidad'] += $cantidad;  // Aumentar cantida
} else {
    $cesta[$producto_id] = array(
        'id' => $producto_id,
        'nombre' => $nombre,
        'precio' => $precio,
        'cantidad' => $cantidad,
    );
}
```

`isset($_SESSION['cesta'])`: Comprueba si ya existe una cesta guardada en la sesión.

- Si existe, la almacena en `$cesta`.
- Si no, inicializa `$cesta` como un array vacío.

Se verifica si el producto ya está en la cesta:

- Si está presente, se incrementa la cantidad del producto.
- Si no está presente, se agrega un nuevo producto a la cesta con los detalles correspondientes (ID, nombre, precio y cantidad).

Finalmente, se actualiza la variable de sesión `$_SESSION['cesta']` con el contenido actualizado de `$cesta`.

Mostrar un mensaje de éxito o error

```php
    $_SESSION['cesta'] = $cesta;  // Actualizar la cesta en la sesión
    echo "<div class='alert alert-success text-center mt-3'>Producto añadido a la cesta.</div>";
} else {
    echo "<div class='alert alert-danger text-center mt-3'>Producto no encontrado.</div>";
}
```

- Si el producto se encontró y se agregó a la cesta, se muestra un mensaje de éxito.
- Si el producto no se encontró, se muestra un mensaje de error

Cerrar la conexión a la base de datos

```php
mysqli_close($conn);  // Cerrar la conexión con la base de datos
```

- Se cierra la conexión con la base de datos usando la función `mysqli_close` para liberar recursos del servidor.

5. db.php

Descripción del código

Este archivo establece la conexión entre tu aplicación web y la base de datos `ferreteria`.
Este script es crucial para todas las páginas de tu aplicación web que necesiten interactuar con la base de datos. Sirve como el punto inicial para:

1. **Conexión**: Establecer una conexión funcional con la base de datos.
2. **Seguridad**: Proporcionar datos de conexión centralizados y reutilizables.
3. **Manejo de Errores**: Detectar problemas de conexión y detener el flujo de la aplicación en caso de fallos.

Datos de conexión

```php
<?php
// Datos de conexión a la base de datos
$host = "localhost";  // Nombre del servidor de la base de datos
$user = "root";       // Nombre de usuario para la base de datos
$password = "";       // Contraseña para la base de datos
$dbname = "ferreteria";  // Nombre de la base de datos
```

`$host`:

- Define el servidor donde está alojada la base de datos. En un servidor local (como XAMPP o WAMP), esto será `"localhost"`.
- Si la base de datos estuviera en un servidor remoto, deberías reemplazar `"localhost"` con la dirección IP o dominio de ese servidor.

`$user`:

- Nombre de usuario que tiene permisos para acceder a la base de datos.
- Por defecto, en XAMPP o WAMP, el usuario principal es `root`.

`$password`:

- Contraseña asociada al usuario. Por defecto, en XAMPP y WAMP, esta contraseña suele estar vacía (`""`).

`$dbname`:

- Nombre de la base de datos a la que deseas conectarte. En este caso, es `ferreteria`.

Crear Conexión con la Base de Datos

```php
// Crear conexión con la base de datos usando mysqli (procedimental)
$conn = mysqli_connect($host, $user, $password, $dbname);
```

1. `mysqli_connect()` es la función procedimental que se utiliza para crear una conexión con la base de datos.
2. Toma como argumentos las variables de conexión (`$host`, `$user`, `$password`, `$dbname`) y devuelve un recurso de conexión que se asigna a la variable `$conn`.
3. Si la conexión es exitosa, `$conn` contendrá un identificador válido de conexión. Si falla, devolverá `false`.

Verificar la Conexión

```php
// Verificar si la conexión ha fallado
if (!$conn) {
    // Mostrar mensaje de error y terminar la ejecución del script
    die("Conexión fallida: " . mysqli_connect_error());
}
?>
```

Comprueba si `$conn` tiene un valor falso (`false`), lo que significa que la conexión ha fallado.

Si falla:

- `mysqli_connect_error()` devuelve un mensaje que describe el error.
- `die()` termina la ejecución del script y muestra el mensaje de error.

6.filtrar_productos.php

Descripción del código

Este archivo permite filtrar productos almacenados en la base de datos basándose en tres criterios principales: nombre, precio mínimo y precio máximo. Proporciona una interfaz moderna con Bootstrap y un flujo de usuario claro, adecuado para explorar la base de datos de manera interactiva.

Encabezado HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Filtrar Productos</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;
        }
    </style>
</head>
```

Estructura del documento:

- `<!DOCTYPE html>`: Define que el archivo utiliza la especificación HTML5.

- `<html lang="es">`: Declara que el idioma principal del contenido es español.
- `<meta charset="UTF-8">`: Especifica la codificación de caracteres como UTF-8 para manejar acentos y caracteres especiales.

Bootstrap CSS:

- Se incluye un enlace al CDN de Bootstrap para que los estilos sean modernos y responsivos.

Estilo personalizado:

- `padding-top: 50px`: Da un espacio en la parte superior para evitar que el contenido quede pegado al borde.

Título e Interfaz de Filtrado

```
<body>
    <div class="container">
        <h1 class="text-center mt-5">Filtrar Productos</h1>
        <div class="row justify-content-center">
            <div class="col-md-8">
                <form action="filtrar_productos.php" method="get" class="form-inline">
                    <div class="form-group mb-2">
                        <label for="nombre" class="mr-2">Nombre:</label>
                        <input type="text" id="nombre" name="nombre" class="form-control">
                    </div>
                    <div class="form-group mx-sm-3 mb-2">
                        <label for="precio_min" class="mr-2">Precio Mínimo:</label>
                        <input type="number" step="0.01" id="precio_min" name="precio_min" class="form-control">
                    </div>
                    <div class="form-group mx-sm-3 mb-2">
                        <label for="precio_max" class="mr-2">Precio Máximo:</label>
                        <input type="number" step="0.01" id="precio_max" name="precio_max" class="form-control">
                    </div>
                    <button type="submit" class="btn btn-primary mb-2">Filtrar</button>
                </form>
            </div>
        </div>
```

Título (`<h1>`):

- Muestra "Filtrar Productos" en el centro de la página con márgenes adicionales hacia arriba (`mt-5`).

Formulario de Filtrado:

- **Método GET**: Envia los datos del formulario a través de la URL, facilitando el análisis de los parámetros recibidos.
- **Campos del formulario**:
  - `nombre`: Permite filtrar productos por nombre.
  - `precio_min`: Permite establecer un precio mínimo.
  - `precio_max`: Permite establecer un precio máximo.
- **Botón de Filtrar**: Envia el formulario al archivo `filtrar_productos.php`.

Conexión y Consulta a la Base de Datos

```php
// Incluir el archivo de conexión a la base de datos
include('db.php');

// Obtener valores del formulario
$nombre = isset($_GET['nombre']) ? $_GET['nombre'] : '';  // Valor del nombre a filtrar
$precio_min = isset($_GET['precio_min']) ? $_GET['precio_min'] : '';  // Valor del precio mínimo a filtrar
$precio_max = isset($_GET['precio_max']) ? $_GET['precio_max'] : '';  // Valor del precio máximo a filtrar
```

Incluir la conexión:

- Se incluye el archivo `db.php` para establecer la conexión con la base de datos.

Captura de parámetros:

- Usa `isset($_GET['nombre'])` para verificar si los valores del formulario han sido enviados.
- Si no se envió un campo, se asigna un valor vacío (`''`).

Construcción de la Consulta SQL

```php
// Construir la consulta SQL para filtrar productos
$sql = "SELECT * FROM productos WHERE 1=1";
if ($nombre != '') {
    $sql .= " AND nombre LIKE '%" . mysqli_real_escape_string($conn, $nombre) . "%'";
}
if ($precio_min != '') {
    $sql .= " AND precio >= " . mysqli_real_escape_string($conn, $precio_min);
}
if ($precio_max != '') {
    $sql .= " AND precio <= " . mysqli_real_escape_string($conn, $precio_max);
}
```

Consulta Base:

- La consulta inicial es `"SELECT * FROM productos WHERE 1=1"`. Esto asegura que siempre haya una base para agregar condiciones adicionales.

Filtros Condicionales:

- Si el campo de `nombre` está definido, añade un filtro con `LIKE` para buscar coincidencias parciales.
- Si `precio_min` o `precio_max` tienen valores, añade filtros para limitar el rango de precios.

Resultados y tabla dinámica

```php
// Ejecutar la consulta
$result = mysqli_query($conn, $sql);

// Verificar si se encontraron productos
if (mysqli_num_rows($result) > 0) {
    // Mostrar los productos en una tabla
    echo "<table class='table table-bordered mt-3'>";
    echo "<thead><tr><th>Nombre</th><th>Descripción</th><th>Precio</th><th>Stock</th></tr
    while ($row = mysqli_fetch_assoc($result)) {
        echo "<tr>
                <td>" . htmlspecialchars($row["nombre"]) . "</td>
                <td>" . htmlspecialchars($row["descripcion"]) . "</td>
                <td>" . htmlspecialchars($row["precio"]) . "</td>
                <td>" . htmlspecialchars($row["stock"]) . "</td>
            </tr>";
    }
    echo "</tbody></table>";
} else {
    // Mostrar mensaje si no se encontraron productos
    echo "<div class='alert alert-warning text-center mt-3'>No se encontraron productos.<
}
```

Verificación de resultados (`mysqli_num_rows`):

- Si la consulta devuelve filas, muestra los productos en una tabla.
- Si no hay resultados, muestra un mensaje de advertencia (`<div class='alert'>`).

Rellenar tabla de resultados:

- `mysqli_fetch_assoc()` recorre cada fila del resultado y la muestra en HTML.
- Se utilizan etiquetas como `<td>` para las columnas de: **Nombre**, **Descripción**, **Precio**, y **Stock**.

Protección contra XSS:

- `htmlspecialchars()` escapa caracteres especiales como `<`, `>` para prevenir inyecciones HTML.

Cierre de la conexión

```php
// Cerrar la conexión con la base de datos
mysqli_close($conn);
?>
```

Libera recursos cerrando la conexión con la base de datos al finalizar la ejecución del script.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

7. Index.php

Descripción de código

Este archivo combina PHP y HTML para crear una página de bienvenida para una ferretería. Tiene funciones relacionadas con la autenticación del usuario y un diseño estilizado para mejorar la experiencia del visitante.

Inicio de Sesión y Autenticación con PHP

```php
<?php
session_start();

// Verificar si las cookies están establecidas
if (isset($_COOKIE['usuario']) && isset($_COOKIE['rol'])) {
    $_SESSION['usuario'] = $_COOKIE['usuario'];
    $_SESSION['rol'] = $_COOKIE['rol'];
}
```

Inicio de sesión con `session_start()`:

- Esto inicia o reanuda una sesión del lado del servidor, permitiendo que variables como `$_SESSION['usuario']` y `$_SESSION['rol']` estén disponibles.

Verificación de cookies:

- Si las cookies `usuario` y `rol` existen, sus valores se asignan a las variables de sesión correspondientes.
- **Propósito**: Implementar la funcionalidad de "recordar sesión" para que un usuario no tenga que iniciar sesión repetidamente mientras las cookies sean válidas.

Estructura HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ferretería</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;
            background: url('https://cdn.pixabay.com/photo/2019/03/29/04/35/tools-4088531_960_720.jpg') no-repeat center center fixed;
            background-size: cover;
            background-blend-mode: overlay;
            background-color: rgba(255, 255, 255, 0.5); /* Ajusta la transparencia aquí */
        }
        .center-text {
            text-align: center;
            margin-top: 20px;
        }
    </style>
</head>
```

HTML básico:

- Define que el documento está en HTML5 (`<!DOCTYPE html>`).
- El idioma del contenido está establecido en español con `lang="es"`.
- Codificación UTF-8 para manejar acentos y caracteres especiales.

Estilo CSS (integrado en la página):

- Se utiliza Bootstrap para un diseño responsivo y moderno.
- **Fondo estilizado**:
  - Incluye una imagen fija como fondo (`background: url(...)`).
  - Mezcla de colores con transparencia (`background-color: rgba(...)`) para darle un efecto claro.
- **Espaciado adicional**:
  - `padding-top: 50px` asegura que el contenido no esté pegado al borde superior.

Cuerpo de la Página

```html
<body>
    <div class="container">
        <h1 class="text-center mt-5">Bienvenidos a la Ferretería</h1>
        <p class="center-text">Tu ferretería de confianza</p>
        <div class="text-center mt-3">
            <a href="login.php" class="btn btn-primary">Iniciar Sesión</a>
        </div>
    </div>
```

Estructura principal con Bootstrap:

- Todo el contenido se encuentra dentro de un contenedor (`<div class="container">`) que aplica márgenes y centrado automático.

Encabezado y Subtítulo:

- `Bienvenidos a la Ferretería`: Texto centrado vertical y horizontalmente.
- `Tu ferretería de confianza`: Mensaje complementario que refuerza la bienvenida.

Botón de inicio de sesión:

- Redirige al usuario a `login.php` para iniciar sesión.
- Diseñado con la clase de Bootstrap `btn btn-primary` para darle un estilo llamativo y moderno.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

8.Inicio.php

Este archivo muestra una lista de productos de una ferretería almacenados en la base de datos, asegurando que solo usuarios autenticados puedan acceder. Además, tiene una interfaz simple y funcional.

Control de Acceso con PHP

```php
<?php
// Iniciar la sesión
session_start();
// Verificar si el usuario no ha iniciado sesión
if (!isset($_SESSION['usuario'])) {
    // Redirigir al formulario de login
    header("Location: login.php");
    exit();
}
?>
```

Iniciar Sesión

- `session_start()`:
  - Esto inicia o reanuda una sesión en el servidor.
  - Permite mantener datos asociados a un usuario durante su navegación, como su estado de autenticación.

Verificar Inicio de Sesión

- `!isset($_SESSION['usuario'])`:

- Comprueba si la variable de sesión `$_SESSION['usuario']` está definida.
- Si no está definida (es decir, el usuario no ha iniciado sesión), el código:
  - Redirige al usuario a la página de login (`login.php`) usando `header("Location: login.php")`.
  - Finaliza la ejecución del script con `exit();`.

Estructura HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ferretería</title>
    <link rel="stylesheet" href="style.css">
</head>
```

Metainformación

- `<!DOCTYPE html>`: Define el uso de HTML5.
- `<html lang="es">`: Declara que el contenido está en español.
- `<meta charset="UTF-8">`: Configura la codificación de caracteres UTF-8 para manejar acentos y caracteres especiales correctamente.

Estilo

- `<link rel="stylesheet" href="style.css">`:
  - Enlaza un archivo externo llamado `style.css`, que contiene los estilos personalizados para la página.

Título y Mensaje de Bienvenida

```html
<body>
    <h1>Bienvenidos a la Ferretería</h1>
```

- `<h1>`: Un encabezado principal que da la bienvenida al usuario con el mensaje "Bienvenidos a la Ferretería".

Consulta a la Base de Datos y Resultados

```php
// Incluir el archivo de conexión a la base de datos
include('db.php');

// Consultar todos los productos en la base de datos
$sql = "SELECT * FROM productos";
$result = mysqli_query($conn, $sql);
```

Conectar a la Base de Datos

- `include('db.php')`:
  - Incluye un archivo externo (`db.php`) que probablemente contiene las credenciales y la conexión a la base de datos.
  - Permite reutilizar el código y mantener la separación de responsabilidades.

Consultar los Productos

- `"SELECT * FROM productos"`:
  - Consulta todos los registros de la tabla `productos` en la base de datos.
  - `$result = mysqli_query($conn,$sql)`:
    - Ejecuta la consulta y almacena el resultado en la variable `$result`.

Mostrar los Resultados

```php
// Verificar si se encontraron productos
if (mysqli_num_rows($result) > 0) {
    // Mostrar los productos en una tabla
    echo "<table>";
    echo "<tr><th>Nombre</th><th>Descripción</th><th>Precio</th></tr>";
    while ($row = mysqli_fetch_assoc($result)) {
        echo "<tr><td>" . htmlspecialchars($row["nombre"]) . "</td><td>" . htmlspecialchars($row["descripcion"]) . "</td><td>" . htmlspecialchars($row["precio"]) . "</td></tr>";
    }
    echo "</table>";
} else {
    echo "No hay productos disponibles.";
}
```

Resultados Encontrados:

- Si hay al menos un producto (`$result->num_rows > 0`), se crea una tabla HTML:
  - `<table>`: Define una tabla.
  - `<tr><th>...</th></tr>`: Encabezado con los nombres de las columnas: "Nombre", "Descripción" y "Precio".
  - `while($row = mysqli_fetch_assoc($result))`:
    - Recorre cada producto en la consulta y lo muestra en una fila (`<tr>`) de la tabla.
    - Los datos de cada producto son extraídos con mysql_fetch_assoc().

Sin Resultados:

- Si no hay productos, muestra un mensaje: `"No hay productos disponibles."`.

Cerrar la Conexión

```php
// Cerrar la conexión con la base de datos
mysqli_close($conn);
```

Finaliza la conexión a la base de datos para liberar los recursos.

9.Introducir_datos.php

Descripción del código

Este archivo está diseñado para permitir que un administrador añada productos a la base de datos de forma segura y sencilla. Incluye autenticación, validación, e interacción con la base de datos.

Encabezado HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Introducir Datos</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;

        }
    </style>
</head>
```

Estructura del documento:

- `<!DOCTYPE html>`: Define el uso de HTML5.
- `<html lang="es">`: Declara que el contenido está en español.
- `<meta charset="UTF-8">`: Configura la codificación como UTF-8, útil para manejar caracteres especiales y acentos correctamente.

Bootstrap CSS:

- Incluye la biblioteca Bootstrap para diseño responsivo y elementos estilizados.

Estilo Personalizado:

- Da un margen superior a todo el contenido del cuerpo (`padding-top: 50px`) para espaciarlo del borde superior de la página.

Encabezado de la Página

```html
<body>
    <div class="container">
        <h1 class="text-center mt-5">Añadir Producto</h1>
```

Encabezado Principal:

- `Añadir Producto` aparece centrado y con un margen superior (`mt-5`) para estar visualmente destacado.

Autenticación de Usuario

```php
// Iniciar la sesión
session_start();

// Verificar si el usuario es administrador
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'administrador') {
    header("Location: login.php");
    exit();
}
```

Inicio de sesión:

- `session_start()` inicia o reanuda la sesión del usuario, permitiendo acceder a las variables de sesión (`$_SESSION`).

Verificación de Administrador:

- Comprueba si el usuario inició sesión (`isset($_SESSION['usuario'])`) y si su rol es `administrador`.
- Si no cumple con estas condiciones, redirige al usuario a la página de inicio de sesión (`login.php`) y detiene el script (`exit();`).
- **Propósito**: Restringir el acceso a esta página solo a administradores.

Manejo del Formulario

```php
// Verificar si el formulario ha sido enviado
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Incluir el archivo de conexión a la base de datos
    include('db.php');

    // Obtener los valores del formulario
    $nombre = $_POST['nombre'];  // Nombre del producto
    $descripcion = $_POST['descripcion'];  // Descripción del producto
    $precio = $_POST['precio'];  // Precio del producto
    $stock = $_POST['stock'];  // Cantidad de stock del producto

    // Escapar datos para evitar inyección de SQL
    $nombre = mysqli_real_escape_string($conn, $nombre);
    $descripcion = mysqli_real_escape_string($conn, $descripcion);
    $precio = mysqli_real_escape_string($conn, $precio);
    $stock = mysqli_real_escape_string($conn, $stock);

    // Insertar los valores en la tabla de productos
    $sql = "INSERT INTO productos (nombre, descripcion, precio, stock) VALUES ('$nombre', '$descripcion', '$precio', '$stock')";

    // Verificar si la inserción ha sido exitosa
    if (mysqli_query($conn, $sql)) {
        echo "<div class='alert alert-success text-center mt-3'>Producto añadido exitosamente.</div>";
    } else {
        // Mostrar mensaje de error si la inserción falla
        echo "<div class='alert alert-danger text-center mt-3'>Error: " . $sql . "<br>" . mysqli_error($conn) . "</div>";
    }

    // Cerrar la conexión con la base de datos
    mysqli_close($conn);
```

Formulario enviado:

- Se verifica si la solicitud es de tipo `POST` para manejar el envío del formulario.

Conexión a la base de datos:

- `include('db.php')` establece la conexión a la base de datos utilizando un archivo externo.

Recoger valores del formulario:

- Los datos del formulario (`nombre`, `descripcion`, `precio`, `stock`) se obtienen de `$_POST`.

Prevención de inyección SQL:

- `mysqli_real_escape_string()` sanitiza los datos recibidos para evitar ataques de inyección SQL.

Consulta SQL de inserción:

- Los valores sanitizados se insertan en la tabla `productos`.

Comprobación de éxito:

- Si la consulta se ejecuta correctamente, muestra un mensaje de éxito.
- En caso de error, se muestra el mensaje devuelto por `mysqli_error($conn)`.

Cerrar conexión:

- `mysqli_close($conn)` libera los recursos al cerrar la conexión con la base de datos.

Formulario de Entrada

```
?>
<div class="row justify-content-center">
    <div class="col-md-6">
        <form action="introducir_datos.php" method="post">
            <div class="form-group">
                <!-- Campo para ingresar el nombre del producto -->
                <label for="nombre">Nombre:</label>
                <input type="text" id="nombre" name="nombre" class="form-control" required>
            </div>
            <div class="form-group">
                <!-- Campo para ingresar la descripción del producto -->
                <label for="descripcion">Descripción:</label>
                <textarea id="descripcion" name="descripcion" class="form-control" required></textarea>
            </div>
            <div class="form-group">
                <!-- Campo para ingresar el precio del producto -->
                <label for="precio">Precio:</label>
                <input type="number" step="0.01" id="precio" name="precio" class="form-control" required>
            </div>
            <div class="form-group">
                <!-- Campo para ingresar el stock del producto -->
                <label for="stock">Stock:</label>
                <input type="number" id="stock" name="stock" class="form-control" required>
            </div>
            <!-- Botón para enviar el formulario -->
            <button type="submit" class="btn btn-primary btn-block">Añadir Producto</button>
        </form>
    </div>
</div>
div>
```

Campos del Formulario

- Campos del Producto:
  - `nombre`: Campo de texto para el nombre del producto.
  - `descripcion`: Área de texto para la descripción.
  - `precio`: Campo numérico para el precio (con decimales).
  - `stock`: Campo numérico para la cantidad en inventario.
- Validación:
  - Cada campo tiene el atributo `required`, lo que obliga al usuario a completarlo antes de enviar el formulario.

Botón de Envío

- Acción del Formulario:
  - Envía los datos a `introducir_datos.php` utilizando el método `POST`.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

10.Leer_datos.php

Descripción del código

El propósito principal es mostrar todos los productos disponibles en la tabla `productos` de la base de datos de manera organizada y estéticamente agradable. Proporciona una interfaz clara para que los usuarios visualicen la información.

Encabezado HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Leer Datos</title>
    <!-- Enlace a Bootstrap CSS -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

</head>
```

Estructura del documento:

- `<!DOCTYPE html>`: Define el uso del estándar HTML5.
- `<html lang="es">`: Declara que el contenido está en español.
- `<meta charset="UTF-8">`: Configura la codificación de caracteres como UTF-8 para soportar caracteres especiales y acentos.

Bootstrap:

- La inclusión de `Bootstrap` mediante un enlace CDN permite un diseño responsivo y atractivo sin necesidad de escribir estilos detallados.

Cuerpo de la Página

```
<body>
    <div class="container">
        <h1 class="text-center mt-5">Lista de Productos</h1>
```

Estructura Principal:

- Todo el contenido está dentro de un contenedor (`<div class="container">`), centrado y con márgenes automáticos proporcionados por Bootstrap.

Título (`<h1>`):

- Muestra el texto "Lista de Productos", centrado con la clase de Bootstrap `text-center`.
- Margen superior adicional (`mt-5`) para separar visualmente el título del borde superior.

Conexión a la Base de Datos

```
// Incluir el archivo de conexión a la base de datos
include('db.php');

// Consultar todos los productos en la base de datos
$sql = "SELECT * FROM productos";
$result = mysqli_query($conn, $sql);
```

- Incluye el archivo `db.php`, que contiene la conexión a la base de datos, utilizando `mysqli_connect` en estilo procedimental.
- Se construye una consulta SQL (`SELECT * FROM productos`) para obtener todos los productos en la tabla `productos`.
- `mysqli_query()` ejecuta la consulta y guarda los resultados en `$result`.

Mostrar Resultados

```
// Verificar si se encontraron productos
if (mysqli_num_rows($result) > 0) {
    echo "<table class='table table-bordered mt-3'>";
    echo "<thead><tr><th>Nombre</th><th>Descripción</th><th>Precio</th><th>Stock</th></tr></thead><tbody>";
    while ($row = mysqli_fetch_assoc($result)) {
        echo "<tr><td>" . htmlspecialchars($row["nombre"]) . "</td><td>" . htmlspecialchars($row["descripcion"]) . "</td><td>" . htmlspecialchars($row["precio"]) . "</td><td>" . htmlspecialchars($row["stock"]) .
    }
    echo "</tbody></table>";
} else {
    echo "<div class='alert alert-warning text-center mt-3'>No hay productos disponibles.</div>";
}
```

- **Comprobar resultados:**
  - `mysqli_num_rows($result)` verifica si la consulta devolvió al menos una fila.
  - Si hay resultados:
    - Muestra una tabla HTML con la clase `table` y `table-bordered` de Bootstrap para que la tabla esté formateada.
  - Cada fila de datos se recorre con `mysqli_fetch_assoc($result)`, que devuelve los datos como un array asociativo.
- **Escapado seguro con `htmlspecialchars`:**
  - Protege las salidas (`$row["nombre"]`, `$row["descripcion"]`, etc.) contra inyecciones XSS (scripts maliciosos) al escapar caracteres especiales.

Estructura final de la tabla:

- Columnas: **Nombre**, **Descripción**, **Precio**, y **Stock**.
- Cada producto aparece en una fila

Si No Hay Resultados

```
} else {
    echo "<div class='alert alert-warning text-center mt-3'>No hay productos disponibles.</div>";
}
```

Cerrar Conexión

```
// Cerrar la conexión con la base de datos
mysqli_close($conn);
?>
```

Finaliza la conexión con la base de datos, liberando recursos.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- **jQuery**: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- **Popper.js**: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- **Bootstrap JS**: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

11. Leer_usuarios.php

Descripción del código

El propósito principal es listar usuarios almacenados en la base de datos, organizándolos según su

rol (clientes y administradores). La interfaz clara y estilizada con Bootstrap facilita la comprensión de los datos.

Encabezado HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Leer Usuarios</title>
    <!-- Enlace a Bootstrap CSS -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
</head>
```

Estructura del Documento:

- `<!DOCTYPE html>`: Define el uso de HTML5.
- `<html lang="es">`: Declara que el contenido está en español.
- `<meta charset="UTF-8">`: Especifica UTF-8 como codificación para manejar acentos y caracteres especiales.

Bootstrap CSS:

- Incluye la biblioteca Bootstrap desde un CDN, facilitando el diseño de la página con estilos predefinidos.

Cuerpo de la Página

```php
<div class="container">
    <h1 class="text-center mt-5">Lista de Usuarios</h1>
    <?php
```

Encabezado:

- `<h1>` Muestra el título principal "Lista de Usuarios".
- La clase `text-center` centra el texto, y `mt-5` añade un margen superior.

Contenedor Principal:

- El contenido está envuelto dentro de un contenedor (`<div class="container">`), lo que garantiza una alineación central y márgenes automáticos para un diseño consistente.

Conexión y Consulta a la Base de Datos

```php
// Incluir el archivo de conexión a la base de datos
include('db.php');

// Consultar todos los usuarios en la base de datos
$sql = "SELECT usuario, rol FROM usuarios";
$result = mysqli_query($conn, $sql);
```

Incluir conexión a la base de datos (`db.php`):

- Asegura que el script pueda interactuar con la base de datos.
- Contiene los datos y configuración necesarios para conectarse.

Consulta de usuarios:

- Ejecuta una consulta SQL para obtener columnas `usuario` y `rol` de la tabla `usuarios`.
- `mysqli_query($conn, $sql)` devuelve el resultado de la consulta.

Clasificación por Roles

```php
// Inicializar arrays para almacenar usuarios por rol
$clientes = array();
$administradores = array();

// Verificar si se encontraron usuarios
if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {
        if ($row["rol"] == "cliente") {
            $clientes[] = $row["usuario"];
        } else if ($row["rol"] == "administrador") {
            $administradores[] = $row["usuario"];
        }
    }
} else {
    echo "<div class='alert alert-warning text-center mt-3'>No hay usuarios disponibles.</div>";
}
```

Inicializar arrays:

- `$clientes` y `$administradores` son listas separadas para agrupar usuarios según su rol.

Procesar resultados:

- Si hay usuarios en la consulta (`mysqli_num_rows` devuelve más de 0), los procesa fila por fila.
- Según el rol del usuario, se agrega su nombre al array correspondiente (`$clientes` o `$administradores`).

Mensaje de no disponible:

- Si no se encuentran resultados, muestra un mensaje de advertencia (`No hay usuarios disponibles`).

Mostrar Usuarios por Rol

```php
// Mostrar usuarios por rol
if (!empty($clientes)) {
    echo "<h2 class='mt-5'>Clientes</h2>";
    echo "<ul class='list-group'>";
    foreach ($clientes as $cliente) {
        echo "<li class='list-group-item'>" . htmlspecialchars($cliente) . "</li>";
    }
    echo "</ul>";
}

if (!empty($administradores)) {
    echo "<h2 class='mt-5'>Administradores</h2>";
    echo "<ul class='list-group'>";
    foreach ($administradores as $admin) {
        echo "<li class='list-group-item'>" . htmlspecialchars($admin) . "</li>";
    }
    echo "</ul>";
}
```

Verificar contenido de los arrays:

- Si `$clientes` no está vacío, muestra un encabezado y crea una lista con nombres.
- Lo mismo ocurre con `$administradores`.

Estructura de listas:

- Usa la clase `list-group` de Bootstrap para mostrar cada rol como una lista estilizada.
- Cada usuario es un elemento (`<li>`) protegido con `htmlspecialchars()` para evitar inyección de HTML.

Cerrar la Conexión

```php
// Cerrar la conexión con la base de datos
mysqli_close($conn);
?>
```

Libera recursos cerrando la conexión con la base de datos.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- **jQuery**: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades

dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

12.Listado_productos.php

Descripción del código

El archivo sirve para listar todos los productos almacenados en la base de datos de una manera estructurada y responsiva. Además:

1. Restringe el acceso a usuarios no autenticados.
2. Añade una funcionalidad exclusiva para clientes (botón "Añadir a la Cesta")

Control de Autenticación (PHP)

```php
<?php
session_start();

// Verificar si el usuario ha iniciado sesión y si su rol es cliente
$esCliente = isset($_SESSION['usuario']) && $_SESSION['rol'] == 'cliente';

if (!$esCliente && !isset($_SESSION['usuario'])) {
    header("Location: login.php");
    exit();
}
```

Inicio de Sesión (`session_start()`):

- Inicia o reanuda una sesión para permitir el uso de variables de sesión (`$_SESSION`).

Verificar Usuario Autenticado:

- `isset($_SESSION['usuario'])`: Comprueba si hay un usuario conectado.
- `$_SESSION['rol'] == 'cliente'`: Verifica si el rol del usuario es "cliente".

Redirección a Login:

- Si el usuario no ha iniciado sesión, lo redirige a la página de inicio de sesión (`login.php`) y detiene la ejecución del script con `exit()`.

Estructura HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Lista de Productos</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;

        }
        .center-text {
            text-align: center;
            margin-top: 20px;
        }
    </style>
</head>
```

Estilo Básico:

- Utiliza Bootstrap para garantizar un diseño responsivo y moderno.
- Incluye un margen superior (`padding-top: 50px`) y centrado de texto (`.center-text`).

Encabezado de la Página

```html
<body>
    <div class="container">
        <h1 class="text-center mt-5">Lista de Productos</h1>
        <div class="text-center mt-3">
            <a href="login.php" class="btn btn-secondary">Cerrar Sesión</a>
        </div>
```

Título Principal:

- "Lista de Productos" aparece centrado con Bootstrap (`text-center`) y separado del borde superior por márgenes (`mt-5`).

Botón de Cerrar Sesión:

- Proporciona un enlace para cerrar sesión redirigiendo a `login.php`.

Conexión y Consulta a la Base de Datos

```php
// Consultar todos los productos en la base de datos
$sql = "SELECT * FROM productos";
$result = mysqli_query($conn, $sql);
```

Conexión:

- `include('db.php')`: Se incluye un archivo que contiene la configuración de la conexión a la base de datos.

Consulta:

- Ejecuta una consulta SQL que selecciona todos los registros de la tabla `productos`.
- Los resultados se almacenan en `$result`.

Mostrar Resultados

```php
// Verificar si se encontraron productos
if (mysqli_num_rows($result) > 0) {
    echo "<table class='table table-bordered mt-3'>";
    echo "<thead><tr><th>Nombre</th><th>Descripción</th><th>Precio</th>";
    if ($esCliente) {
        echo "<th>Acciones</th>";
    }
    echo "</tr></thead><tbody>";
    while ($row = mysqli_fetch_assoc($result)) {
        echo "<tr>
                <td>" . htmlspecialchars($row["nombre"]) . "</td>
                <td>" . htmlspecialchars($row["descripcion"]) . "</td>
                <td>" . htmlspecialchars($row["precio"]) . "</td>";
        if ($esCliente) {
            echo "<td>
                    <!-- Botón para agregar el producto a la cesta -->
                    <a href='añadir_cesta.php?id=" . $row["id"] . "' class='btn btn-primary btn-sm'>Añadir a la Cesta</a>
                  </td>";
        }
        echo "</tr>";
    }
    echo "</tbody></table>";
} else {
    echo "<div class='alert alert-warning text-center mt-3'>No hay productos disponibles.</div>";
}
```

Condicional para resultados:

- Si `mysqli_num_rows($result)` devuelve más de 0, muestra los productos en una tabla.
- Si no hay resultados, muestra un mensaje de advertencia.

Tabla HTML:

- `table table-bordered`: Clase de Bootstrap que añade bordes y diseño a la tabla.
- Cada fila incluye el **nombre**, **descripción** y **precio** del producto.

Opciones para cliente:

- Si `$esCliente` es verdadero, muestra una columna adicional con acciones específicas.
- Por ejemplo, un botón que redirige a `añadir_cesta.php` para agregar productos a la

cesta.

Sanitización:

- Usa `htmlspecialchars()` para prevenir ataques XSS al imprimir datos desde la base de datos.

Cerrar Conexión

```php
// Cerrar la conexión con la base de datos
mysqli_close($conn);
?>
```

Libera recursos al cerrar la conexión con la base de datos.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

13. Login.php

Descripción del código

Este archivo implementa un sistema seguro para autenticación de usuarios:

1. Recoge datos del formulario de inicio de sesión.
2. Valida las credenciales en la base de datos.
3. Establece sesiones/cookies y redirige al usuario según su rol.

Encabezado HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;
        }
        .usuario-nuevo {
            text-align: center;
            margin-top: 20px;
        }
    </style>
</head>
```

Metainformación:

- El idioma está definido como español (`<html lang="es">`).
- La codificación UTF-8 se utiliza para manejar acentos y caracteres especiales (`<meta charset="UTF-8">`).

Bootstrap y Estilo Personalizado:

- Incluye Bootstrap desde un CDN para un diseño responsivo.
- El espaciado superior (`padding-top: 50px`) y centrado del texto se personalizan mediante CSS.

Cuerpo e Interfaz del Formulario

```html
<body>
    <div class="container">
        <h1 class="text-center mt-5">Iniciar Sesión</h1>
        <div class="row justify-content-center">
            <div class="col-md-6">
                <form action="login.php" method="post">
                    <div class="form-group">
                        <!-- Campo para ingresar el nombre de usuario -->
                        <label for="usuario">Usuario:</label>
                        <input type="text" id="usuario" name="usuario" class="form-control" required>
                    </div>
                    <div class="form-group">
                        <!-- Campo para ingresar la contraseña -->
                        <label for="password">Contraseña:</label>
                        <input type="password" id="password" name="password" class="form-control" required>
                    </div>
                    <div class="form-group form-check">
                        <!-- Checkbox para recordar al usuario -->
                        <input type="checkbox" class="form-check-input" id="recordar" name="recordar">
                        <label class="form-check-label" for="recordar">Recordarme</label>
                    </div>
                    <!-- Botón para enviar el formulario -->
                    <button type="submit" class="btn btn-primary btn-block">Iniciar Sesión</button>
                </form>
            </div>
        </div>
        <!-- Frase de Usuario Nuevo y botón de registro -->
        <div class="usuario-nuevo">
            <p>¿Usuario nuevo?</p>
            <a href="register.php" class="btn btn-secondary">Registrarse</a>
        </div>
    </div>
```

Estructura y Diseño del Formulario:

- Incluye campos para usuario, contraseña y un checkbox opcional para "Recordarme".
- Un botón de envío (`type="submit"`) que envía los datos a `login.php` utilizando el método `POST`.

Sección "Usuario Nuevo":

- Invita a nuevos usuarios a registrarse a través de un enlace que redirige a `register.php`.

Procesar el Inicio de Sesión (PHP)

```php
// Verificar si el formulario ha sido enviado
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Incluir el archivo de conexión a la base de datos
    include('db.php');

    // Obtener los valores del formulario
    $usuario = mysqli_real_escape_string($conn, $_POST['usuario']);
    $password = $_POST['password'];

    // Buscar al usuario en la base de datos
    $sql = "SELECT * FROM usuarios WHERE usuario='$usuario'";
    $result = mysqli_query($conn, $sql);

    // Verificar si se encontró el usuario
    if (mysqli_num_rows($result) > 0) {
        // Obtener los datos del usuario
        $row = mysqli_fetch_assoc($result);

        // Verificar si la contraseña es correcta
        if (password_verify($password, $row['password'])) {
            // Almacenar datos del usuario en la sesión
            $_SESSION['usuario'] = $row['usuario'];
            $_SESSION['rol'] = $row['rol'];

            // Verificar si se debe recordar al usuario
            if (isset($_POST['recordar'])) {
                // Establecer cookies para recordar al usuario por 30 días
                setcookie('usuario', $row['usuario'], time() + (86400 * 30), "/");
                setcookie('rol', $row['rol'], time() + (86400 * 30), "/");
            }

            // Redirigir según el rol del usuario
            if ($row['rol'] == 'administrador') {
                header("Location: admin_menu.php");
            } else {
                header("Location: listado_productos.php");
            }
        } else {
            // Mostrar mensaje de error si la contraseña es incorrecta
            echo "<div class='alert alert-danger text-center mt-3'>Contraseña incorrecta.</div>";
        }
    } else {
        // Mostrar mensaje de error si no se encuentra al usuario
        echo "<div class='alert alert-danger text-center mt-3'>Usuario no encontrado.</div>";
    }
```

Inicio de sesión:

- `session_start()` habilita el uso de variables de sesión.

Procesar formulario:

- Verifica si la solicitud es de tipo POST.
- Incluye el archivo de conexión (`db.php`) para interactuar con la base de datos.
- Los valores de usuario y contraseña se obtienen del formulario.

Consulta SQL:

- Busca en la base de datos al usuario ingresado.
- Escapa los datos del usuario (`mysqli_real_escape_string`) para evitar inyecciones SQL.

Validación del usuario:

- Comprueba si el usuario existe (`mysqli_num_rows`).
- Si existe, compara la contraseña ingresada con la encriptada en la base de datos mediante `password_verify`.
- Si la contraseña es válida:
  - Guarda datos importantes en la sesión (`$_SESSION`).
  - Crea cookies si el checkbox "Recordarme" fue marcado (`setcookie`).
  - Redirige al usuario según su rol (administrador o cliente).

Mensajes de error:

- Si la contraseña es incorrecta o el usuario no existe, muestra mensajes de error personalizados.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

14. logout.php

Descripción del código

Este script es esencial para un flujo seguro de cierre de sesión en aplicaciones web. Su objetivo principal es:

1. Finalizar cualquier sesión activa del usuario.
2. Asegurar que el usuario no pueda seguir accediendo a secciones protegidas de la aplicación tras el cierre de sesión.
3. Redirigir al usuario al formulario de inicio de sesión para una nueva autenticación.

Iniciar la Sesión

```php
<?php
// Iniciar la sesión
session_start();
```

Este comando asegura que la sesión actual del usuario se active. Si no hay una sesión en curso, `session_start()` intenta iniciarla.
Propósito: Antes de destruir cualquier dato relacionado con la sesión, necesitas asegurarte de que la sesión exista o esté activa.

Destruir las Variables de Sesión

```php
// Destruir todas las variables de sesión
session_destroy();
```

Qué hace:

- Esta función destruye todos los datos asociados con la sesión actual.
- Cierra la sesión en el servidor, eliminando las variables almacenadas.

Importante: Aunque destruye la sesión en el servidor, no elimina automáticamente las cookies de sesión en el cliente.

Redirigir al Formulario de Inicio de Sesión

```php
// Redirigir al formulario de login
header("Location: login.php");
exit();
```

Redirección:

- El usuario es redirigido a `login.php`, típicamente el formulario de inicio de sesión.

Por qué usar `exit()`:

- Garantiza que el script detenga su ejecución inmediatamente después de redirigir.

15. Modificar_datos.php

Descripción del código

Esta página permite a los administradores:

1. Seleccionar un producto de la lista.
2. Modificar sus datos (nombre, descripción, precio, stock).
3. Ver un listado de usuarios(Clientes y administradores).
4. Eliminar productos de manera segura y directa.

Encabezado HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Modificar Datos</title>
    <!-- Enlace a Bootstrap CSS -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
</head>
```

Metainformación:

- Define el uso del estándar HTML5 (`<!DOCTYPE html>`).
- El idioma está configurado como español (`<html lang="es">`).
- La codificación UTF-8 asegura el manejo correcto de caracteres especiales y acentos (`<meta charset="UTF-8">`).

Bootstrap CSS:

- Incluye Bootstrap para proporcionar un diseño responsivo y moderno.

Verificación de Sesión (PHP)

```php
// Iniciar la sesión
session_start();

// Verificar si el usuario es administrador
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'administrador') {
    header("Location: login.php");
    exit();
}
```

Inicio de Sesión:

- `session_start()` inicia o reanuda una sesión activa.

Restricción por Rol:

- Verifica si el usuario está autenticado y si su rol es "administrador".
- Si no cumple las condiciones, redirige al usuario a `login.php` y detiene la ejecución del script.

Conexión y Operaciones en la Base de Datos

Modificar Producto

```php
// Verificar si se ha enviado el formulario para modificar un producto
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['modificar'])) {
    $id = $_POST['id'];
    $nombre = mysqli_real_escape_string($conn, $_POST['nombre']);
    $descripcion = mysqli_real_escape_string($conn, $_POST['descripcion']);
    $precio = mysqli_real_escape_string($conn, $_POST['precio']);
    $stock = mysqli_real_escape_string($conn, $_POST['stock']);

    // Actualizar los valores del producto en la base de datos
    $sql = "UPDATE productos SET nombre='$nombre', descripcion='$descripcion', precio='$precio', stock='$stock' WHERE id=$id";

    // Verificar si la actualización ha sido exitosa
    if (mysqli_query($conn, $sql)) {
        echo "<div class='alert alert-success text-center mt-3'>Producto modificado exitosamente.</div>";
    } else {
        echo "<div class='alert alert-danger text-center mt-3'>Error: " . $sql . "<br>" . mysqli_error($conn) . "</div>";
    }
}
```

Eliminar Producto

```php
// Verificar si se ha enviado el formulario para eliminar un producto
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['eliminar'])) {
    $id = $_POST['id'];

    // Eliminar el producto de la base de datos
    $sql = "DELETE FROM productos WHERE id=$id";

    // Verificar si la eliminación ha sido exitosa
    if (mysqli_query($conn, $sql)) {
        echo "<div class='alert alert-success text-center mt-3'>Producto eliminado exitosamente.</div>";
    } else {
        echo "<div class='alert alert-danger text-center mt-3'>Error: " . $sql . "<br>" . mysqli_error($conn) . "</div>";
    }
}

// Verificar si se ha enviado el formulario para seleccionar un producto
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['seleccionar'])) {
    $id = $_POST['id'];
    $sql = "SELECT * FROM productos WHERE id=$id";
    $result = mysqli_query($conn, $sql);
    $producto = mysqli_fetch_assoc($result);
}
?>
```

Seleccionar Producto

```php
// Verificar si se ha enviado el formulario para seleccionar un producto
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['seleccionar'])) {
    $id = $_POST['id'];
    $sql = "SELECT * FROM productos WHERE id=$id";
    $result = mysqli_query($conn, $sql);
    $producto = mysqli_fetch_assoc($result);
}
?>
```

Verificar tipo de solicitud:

- Se asegura de que el formulario enviado sea de tipo `POST`.

Modificaciones y sanitización:

- Los datos del formulario se procesan y se limpian con `mysqli_real_escape_string` para evitar inyecciones SQL.

Modificar producto:

- Actualiza el producto especificado en la base de datos.
- Muestra mensajes de éxito o error.

Eliminar producto:

- Borra el producto de la base de datos según el ID proporcionado.

Seleccionar producto:

- Recupera un producto específico de la base de datos para mostrarlo en un formulario editable.

Formularios HTML

Seleccionar Producto

```php
<!-- Formulario para seleccionar un producto -->
<form action="modificar_datos.php" method="post" class="mt-3">
    <div class="form-group">
        <label for="id">Seleccionar Producto:</label>
        <select id="id" name="id" class="form-control" required>
            <?php
            // Consultar todos los productos en la base de datos
            $sql = "SELECT id, nombre FROM productos";
            $result = mysqli_query($conn, $sql);
            while ($row = mysqli_fetch_assoc($result)) {
                echo "<option value='" . $row["id"] . "'>" . htmlspecialchars($row["nombre"]) . "</option>";
            }
            ?>
        </select>
    </div>
    <button type="submit" name="seleccionar" class="btn btn-primary btn-block">Seleccionar</button>
    <a href="admin_menu.php" class="btn btn-primary btn-block">Volver</a>
</form>
```

Lista desplegable:

- Recupera todos los productos desde la base de datos y los muestra en un menú desplegable.

Botones:

- "Seleccionar" envía el formulario y recupera el producto elegido.
- "Volver" redirige al menú del administrador

Modificar o Eliminar Producto

```php
<!-- Formulario para modificar un producto -->
<?php if (isset($producto)) { ?>
<form action="modificar_datos.php" method="post" class="mt-3">
    <input type="hidden" name="id" value="<?php echo $producto['id']; ?>">
    <div class="form-group">
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre" class="form-control" value="<?php echo htmlspecialchars($producto['nombre']); ?>" required>
    </div>
    <div class="form-group">
        <label for="descripcion">Descripción:</label>
        <textarea id="descripcion" name="descripcion" class="form-control" required><?php echo htmlspecialchars($producto['descripcion']); ?></textarea>
    </div>
    <div class="form-group">
        <label for="precio">Precio:</label>
        <input type="number" step="0.01" id="precio" name="precio" class="form-control" value="<?php echo htmlspecialchars($producto['precio']); ?>" required>
    </div>
    <div class="form-group">
        <label for="stock">Stock:</label>
        <input type="number" id="stock" name="stock" class="form-control" value="<?php echo htmlspecialchars($producto['stock']); ?>" required>
    </div>
    <button type="submit" name="modificar" class="btn btn-primary btn-block">Modificar Producto</button>
    <button type="submit" name="eliminar" class="btn btn-danger btn-block">Eliminar Producto</button>
</form>
<?php } ?>
</div>
```

Si un producto ha sido seleccionado:

1. Crea un formulario con los datos del producto, permitiendo editarlos o eliminarlos.
2. Los valores están pre-rellenados con datos del producto

Script para interactividad

```php
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- **jQuery**: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- **Popper.js**: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- **Bootstrap JS**: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

16. Register.php

Descripción del código

El objetivo principal es permitir que nuevos usuarios se registren en el sistema con:

1. Almacenamiento seguro de contraseñas encriptadas.
2. Diferenciación entre roles (cliente y administrador).
3. Mensajes de feedback dependiendo del resultado del registro.

Encabezado HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Registro</title>
    <!-- Enlace a Bootstrap CSS -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
</head>
```

Estructura Base:

- `<!DOCTYPE html>`: Define el uso de HTML5.
- `<html lang="es">`: Configura el idioma del contenido como español.
- `<meta charset="UTF-8">`: Utiliza UTF-8 para manejar correctamente caracteres como acentos y ñ.

Estilos (Bootstrap):

- La inclusión de Bootstrap a través de un CDN facilita el diseño responsivo y moderno.

Estructura de la Página y del Formulario

```html
    <!-- Campo para ingresar el nombre de usuario -->
    <label for="usuario">Usuario:</label>
    <input type="text" id="usuario" name="usuario" class="form-control" required>
    <label for="apellido1">Apellido1:</label>
    <input type="text" id="apellido1" name="apellido1" class="form-control" required>
    <label for="apellido2">Apellido2:</label>
    <input type="text" id="apellido2" name="apellido2" class="form-control" required>
</div>
<div class="form-group">
    <!-- Campo para ingresar el email -->
    <label for="email">Correo Electrónico:</label>
    <input type="email" id="email" name="email" class="form-control" required>
</div>
<div class="form-group">
    <!-- Campo para ingresar la contraseña -->
    <label for="password">Contraseña:</label>
    <input type="password" id="password" name="password" class="form-control" required>
</div>
<div class="form-group">
    <!-- Selección del rol del usuario -->
    <label for="rol">Rol:</label>
    <select id="rol" name="rol" class="form-control">
        <option value="cliente">Cliente</option>
        <option value="administrador">Administrador</option>
    </select>
</div>
<!-- Botón para enviar el formulario -->
<button type="submit" class="btn btn-primary btn-block">Registrar</button>
<a href="login.php" class="btn btn-primary btn-block">Iniciar Sesión</a>
form>
```

Formulario Principal:

- Campos de entrada para `usuario`, `apellido1`, `apellido2`, contraseña y selección del rol.
- Cada campo tiene el atributo `required` para forzar que los usuarios llenen todos los campos antes de enviar el formulario.

Rol del Usuario:

- El campo `<select>` permite elegir entre "Cliente" o "Administrador".

Botones:

- **Registrar**: Envía los datos al script `register.php` mediante el método `POST`.
- **Iniciar Sesión**: Redirige al usuario al formulario de login.

Manejo del Registro con PHP

```php
<?php
// Verificar si el formulario ha sido enviado
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Incluir el archivo de conexión a la base de datos
    include('db.php');

    // Obtener los valores del formulario y sanitizarlos
    $usuario = mysqli_real_escape_string($conn, $_POST['usuario']);
    $apellido1 = mysqli_real_escape_string($conn, $_POST['apellido1']);
    $apellido2 = mysqli_real_escape_string($conn, $_POST['apellido2']);
    $email = mysqli_real_escape_string($conn, $_POST['email']); // Procesar el campo email
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT); // Encriptar la contraseña
    $rol = mysqli_real_escape_string($conn, $_POST['rol']);

    // Validar que el email tenga un formato válido
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "<div class='alert alert-danger text-center mt-3'>El formato del correo electrónico no es v
    } else {
        // Si el rol es administrador, el campo 'validado' será 0 por defecto
        $validado = ($rol == 'administrador') ? 0 : 1;

        // Insertar los valores en la tabla de usuarios
        $sql = "INSERT INTO usuarios (usuario, apellido1, apellido2, email, password, rol, validado)
                VALUES ('$usuario', '$apellido1', '$apellido2', '$email', '$password', '$rol', $validado

        // Verificar si la inserción ha sido exitosa
        if (mysqli_query($conn, $sql)) {
            if ($rol == 'administrador') {
                echo "<div class='alert alert-warning text-center mt-3'>Registro exitoso. Espera validac
            } else {
                echo "<div class='alert alert-success text-center mt-3'>Registro exitoso.</div>";
            }
        } else {
            echo "<div class='alert alert-danger text-center mt-3'>Error: " . mysqli_error($conn) . "</d
        }
    }

    // Cerrar la conexión con la base de datos
    mysqli_close($conn);
}
```

Validación de solicitud:

- Comprueba si el formulario fue enviado mediante el método `POST`.

Sanitización de datos:

- Usa `mysqli_real_escape_string` para proteger los datos del formulario contra inyección SQL.
- Encripta la contraseña con `password_hash` para almacenamiento seguro.

Validación según rol:

- Los usuarios con rol `administrador` requieren validación manual (`validado = 0`).
- Los usuarios `cliente` están validados automáticamente (`validado = 1`).

Consulta SQL:

- Inserta los datos en la tabla `usuarios`.
- Muestra mensajes de éxito o error según el resultado de la consulta.

Cierre de conexión:

- Libera los recursos del servidor con `mysqli_close`.

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

17. Style.css

Selección Global del Body

```
body {
    font-family: Arial, sans-serif;
    background: url('...') no-repeat center center fixed;
}
```

`font-family: Arial, sans-serif;`:

- Establece la fuente global del documento, con Arial como la fuente principal y `sans-serif` como respaldo.

`background: url('...') no-repeat center center fixed;`:

- Asigna una imagen de fondo que:

- `no-repeat`: No se repite en mosaico.
- `center center`: Se centra tanto horizontal como verticalmente.
- `fixed`: Se fija al fondo de la pantalla y no se desplaza al hacer scroll.

Estilo para el Encabezado `h1`

```css
h1 {
    text-align: center;
    margin-top: 20px;
}
```

`text-align: center;`:

- Centra el texto del encabezado horizontalmente.

`margin-top: 20px;`:

- Crea un espacio superior de 20px para separar visualmente el encabezado de otros elementos.

Configuración de Contenedores

```css
.container {
    margin-top: 50px;
}
```

`margin-top: 50px;`:

- Aplica un margen superior a los contenedores para garantizar que el contenido no esté pegado al borde superior de la ventana.

Estilo para Alerta (`.alert`)

```css
.alert {
    margin-top: 20px;
}
```

`margin-top: 20px;`:

- Añade un espacio superior de 20px a las alertas, lo que mejora la separación entre elementos en la interfaz.

18. Validar.php

Inicio de Sesión y Verificación

```php
<?php
// Iniciar la sesión
session_start();

// Verificar si el usuario tiene rol de administrador
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'administrador') {
    header("Location: login.php");
    exit();
}
```

Propósito:

- Este bloque asegura que solo los usuarios autenticados como "administrador" tengan acceso al archivo.

Explicación:

- `session_start()`: Reanuda o inicia una sesión para acceder a las variables `$_SESSION`.
- Verificación de permisos:
  - Comprueba si la sesión contiene un `usuario` activo y si su `rol` es `administrador`.
  - Si no se cumple la condición, se redirige al usuario a la página de inicio de sesión (`login.php`) y detiene la ejecución con `exit()`.

Conexión a la Base de Datos

```php
// Incluir el archivo de conexión a la base de datos
include('db.php');
```

Propósito:

- Incluir un archivo que configure la conexión a la base de datos mediante la variable `$conn`.

Requisitos:

- El archivo `db.php` debe contener los detalles del servidor y la base de datos.

Lógica para Aprobar o Rechazar Usuarios

```php
// Aprobar o rechazar solicitudes de administrador
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['aprobar'])) {
        $usuario_id = intval($_POST['usuario_id']); // Convertir a entero por seguridad
        $sql = "UPDATE usuarios SET validado = 1 WHERE id = ?";
        $stmt = mysqli_prepare($conn, $sql);
        mysqli_stmt_bind_param($stmt, "i", $usuario_id);
        mysqli_stmt_execute($stmt);
        mysqli_stmt_close($stmt);
        header("Location: validar.php?status=aprobado");
        exit();
    } elseif (isset($_POST['rechazar'])) {
        $usuario_id = intval($_POST['usuario_id']);
        $sql = "DELETE FROM usuarios WHERE id = ?";
        $stmt = mysqli_prepare($conn, $sql);
        mysqli_stmt_bind_param($stmt, "i", $usuario_id);
        mysqli_stmt_execute($stmt);
        mysqli_stmt_close($stmt);
        header("Location: validar.php?status=rechazado");
        exit();
    }
}
```

Solicitudes POST:

- Comprueba si la solicitud es de tipo `POST`, lo cual ocurre al enviar formularios para aprobar o rechazar.

Aprobar administrador:

- Actualiza el campo `validado` a 1 (aprobado) en la tabla `usuarios` para el administrador seleccionado.

Rechazar administrador:

- Elimina de la base de datos al administrador rechazado.

Consultas seguras:

- Se utilizan **consultas preparadas** (`mysqli_prepare`) para evitar inyecciones SQL.
- `mysqli_stmt_bind_param` asegura que el ID del usuario sea tratado como un entero (`i`).

Redirección:

- Después de procesar la solicitud, redirige a `validar.php` con el estado de la acción (`aprobado` o `rechazado`).

Consulta de Administradores Pendientes

```php
// Consultar administradores pendientes
$sql = "SELECT id, usuario FROM usuarios WHERE rol = 'administrador' AND validado = 0";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
```

Propósito:

- Recuperar la lista de administradores pendientes de validación.

Explicación:

- Filtra a los usuarios con el rol de `administrador` y `validado = 0`, es decir, administradores que todavía no han sido aprobados.

Interfaz para Aprobar o Rechazar Administradores

```php
// Verificar si hay solicitudes pendientes
if ($result->num_rows > 0) {
    echo "<table class='table table-bordered mt-3'>";
    echo "<thead><tr><th>Usuario</th><th>Acciones</th></tr></thead><tbody>";
    while ($row = $result->fetch_assoc()) {
        echo "<tr>
            <td>{$row['usuario']}</td>
            <td>
                <form action='validar.php' method='post' style='display:inline;'>
                    <input type='hidden' name='usuario_id' value='{$row['id']}'>
                    <button type='submit' name='aprobar' class='btn btn-success btn-sm'>Aprobar</button>
                </form>

                <form action='validar.php' method='post' style='display:inline;'>
                    <input type='hidden' name='usuario_id' value='{$row['id']}'>
                    <button type='submit' name='rechazar' class='btn btn-danger btn-sm'>Rechazar</button>
                </form>
            </td>
        </tr>";
    }
    echo "</tbody></table>";
} else {
    echo "<div class='alert alert-warning text-center mt-3'>No hay solicitudes de administrador pendientes.</div>";
}
?>
```

Propósito:

- Generar dinámicamente una tabla con los administradores pendientes, incluyendo botones para "Validar" y "Rechazar".

Explicación:

- Recorre los resultados de la consulta con `$result->fetch_assoc()` para mostrar cada administrador en una fila de la tabla.
- Para cada administrador:
    - **Formulario de "Aprobar"**:
        - Envía `usuario_id` al archivo `validar.php` con el botón `aprobar`.
    - **Formulario de "Rechazar"**:
        - Envía `usuario_id` al archivo `validar.php` con el botón `rechazar`.

Mostrar solicitudes pendientes

```php
// Verificar si hay solicitudes pendientes
if (mysqli_num_rows($result) > 0) {
    echo "<table class='table table-bordered mt-3'>";
    echo "<thead><tr><th>Usuario</th><th>Acciones</th></tr></thead><tbody>";
    while ($row = mysqli_fetch_assoc($result)) {
        echo "<tr>
            <td>" . htmlspecialchars($row['usuario']) . "</td>
            <td>
                <form action='validar.php' method='post' style='display:inline;'>
                    <input type='hidden' name='usuario_id' value='" . htmlspecialchars($row['id']) . "'>
                    <button type='submit' name='aprobar' class='btn btn-success btn-sm'>Aprobar</button>
                </form>

                <form action='validar.php' method='post' style='display:inline;'>
                    <input type='hidden' name='usuario_id' value='" . htmlspecialchars($row['id']) . "'>
                    <button type='submit' name='rechazar' class='btn btn-danger btn-sm'>Rechazar</button>
                </form>
            </td>
        </tr>";
    }
    echo "</tbody></table>";
} else {
    echo "<div class='alert alert-warning text-center mt-3'>No hay solicitudes de administrador pendientes.</div>";
}
```

Verificar solicitudes:

- Si hay registros pendientes (`mysqli_num_rows > 0`), los muestra en una tabla.
- Si no hay registros, muestra un mensaje de advertencia.

Tabla de solicitudes:

- Incluye el nombre de usuario y botones para aprobar o rechazar.

Botones de acciones:

- Cada fila tiene dos formularios:
  - Uno para aprobar (`name='aprobar'`).
  - Otro para rechazar (`name='rechazar'`).
- Los valores son protegidos con `htmlspecialchars()` para evitar ataques XSS.

Mensajes de Confirmación

```php
// Mensajes de estado después de aprobar/rechazar
if (isset($_GET['status'])) {
    if ($_GET['status'] == 'aprobado') {
        echo "<div class='alert alert-success text-center mt-3'>Administrador aprobado correctamente.</div>";
    } elseif ($_GET['status'] == 'rechazado') {
        echo "<div class='alert alert-success text-center mt-3'>Administrador rechazado correctamente.</div>";
    }
}
```

Propósito:

- Mostrar mensajes claros después de aprobar o rechazar una solicitud.

Explicación:

- Si la URL contiene el parámetro `status=aprobado`, se muestra un mensaje de éxito para la aprobación.
- Similar para `status=rechazado`.

Cierre de recursos

```php
// Cerrar el statement y conexión
mysqli_stmt_close($stmt);
mysqli_close($conn);
```

Libera los recursos del `statement` y cierra la conexión a la base de datos para optimizar el servidor.

Enlace para Regresar

```php
<div class="text-center mt-3">
    <a href="admin_menu.php" class="btn btn-primary">Volver al Menú del Administrador</a>
</div>
```

Propósito:

- Permitir a los administradores volver al menú principal después de realizar una acción.

19. Ver_cesta.php

Descripción del código

Este archivo PHP implementa la funcionalidad de un carrito de compras para clientes. Permite al usuario ver los productos en su cesta, actualizar cantidades, eliminar productos y proceder al pago.

Inicio de Sesión y Verificación del Cliente

```php
<?php
session_start();
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'cliente') {
    header("Location: login.php");
    exit();
}
```

Propósito:

- Asegura que solo los clientes registrados puedan acceder a esta página.
- Redirige a `login.php` si el usuario no ha iniciado sesión o si no tiene el rol `cliente`.

Manejo de la Cesta

Obtener la Cesta del Cliente

```php
// Obtener la cesta del cliente desde la sesión
$cesta = isset($_SESSION['cesta']) ? $_SESSION['cesta'] : [];
```

Propósito:

- Recuperar los datos del carrito de la sesión. Si no existe una cesta, inicializa un array vacío.

Actualizar Cantidades

```php
// Actualizar la cantidad de productos en la cesta
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['actualizar'])) {
    $producto_id = $_POST['producto_id'];  // ID del producto a actualizar
    $nueva_cantidad = $_POST['cantidad'];  // Nueva cantidad del producto
    if ($nueva_cantidad > 0) {
        $cesta[$producto_id]['cantidad'] = $nueva_cantidad;
    } else {
        unset($cesta[$producto_id]);  // Eliminar producto si la cantidad es 0
    }
    $_SESSION['cesta'] = $cesta;  // Actualizar la cesta en la sesión
}
```

Propósito:

- Permitir al cliente modificar la cantidad de un producto en el carrito.
- Si la cantidad es `0`, elimina el producto de la cesta.
- Actualiza los datos en la sesión.

Eliminar Producto

```php
// Eliminar un producto de la cesta
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['eliminar'])) {
    $producto_id = $_POST['producto_id'];  // ID del producto a eliminar
    unset($cesta[$producto_id]);  // Eliminar el producto de la cesta
    $_SESSION['cesta'] = $cesta;  // Actualizar la cesta en la sesión
}
```

Propósito:

- Permitir al cliente eliminar un producto del carrito.
- Actualiza los datos en la sesión.

Interfaz: Mostrar Cesta

Estructura Principal

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ver Cesta</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;
            background: url('https://cdn.pixabay.com/photo/2019/03/29/04/35/tools-4088531_960_720.jpg') no-repeat center center fixed;
            background-size: cover;
            background-blend-mode: overlay;
            background-color: rgba(255, 255, 255, 0.5); /* Ajusta la transparencia aquí */

        }
    </style>
</head>
```

Propósito:

- Incluye Bootstrap para crear un diseño responsivo y utiliza estilos personalizados para mejorar la apariencia.

Mostrar Productos en la Cesta

```php
    if (empty($cesta)) {
        echo "<div class='alert alert-warning text-center mt-3'>Tu cesta está vacía.</div>";
    } else {
        echo "<table class='table table-bordered mt-3'>";
        echo "<thead><tr><th>Producto</th><th>Cantidad</th><th>Precio Unitario</th><th>Total</th><th>Acciones</th></tr></thead><tbody>";
        $total = 0;
        foreach ($cesta as $producto_id => $producto) {
            $subtotal = $producto['cantidad'] * $producto['precio'];
            $total += $subtotal;
            echo "<tr>
                <td>{$producto['nombre']}</td>
                <td>
                    <form action='ver_cesta.php' method='post' style='display:inline;'>
                        <input type='hidden' name='producto_id' value='{$producto_id}'>
                        <input type='number' name='cantidad' value='{$producto['cantidad']}' min='1' class='form-control d-inline' style='width: 70px;'>
                        <button type='submit' name='actualizar' class='btn btn-primary btn-sm'>Actualizar</button>
                    </form>
                </td>
                <td>{$producto['precio']}</td>
                <td>{$subtotal}</td>
                <td>
                    <form action='ver_cesta.php' method='post' style='display:inline;'>
                        <input type='hidden' name='producto_id' value='{$producto_id}'>
                        <button type='submit' name='eliminar' class='btn btn-danger btn-sm'>Eliminar</button>
                    </form>
                </td>
            </tr>";
        }
        echo "<tr><td colspan='3' class='text-right'><strong>Total</strong></td><td colspan='2'><strong>{$total}</strong></td></tr>";
        echo "</tbody></table>";
    }
```

Propósito:

- Muestra los productos en la cesta en una tabla con las siguientes columnas:
    - **Producto**: Nombre del producto.
    - **Cantidad**: Formulario para modificar la cantidad.
    - **Precio Unitario**: Precio de un solo producto.
    - **Total**: Precio total para ese producto (`cantidad × precio`).
    - **Acciones**:
        - Botones para actualizar la cantidad o eliminar el producto.

Acciones: Seguir Comprando y Confirmar Compra

```
<div class="text-center mt-3">
    <a href="listado_productos.php" class="btn btn-primary">Seguir Comprando</a>
    <a href="confirmar_cesta.php" class="btn btn-success">Confirmar Compra</a>
</div>
```

- **Propósito**:
    - Permite al cliente:
        - Volver a la lista de productos (`listado_productos.php`) para añadir más productos.
        - Proceder a confirmar su compra (`confirmar_cesta.php`).

Script para interactividad

```
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

- jQuery: Biblioteca JavaScript que facilita la manipulación del DOM y añade funcionalidades dinámicas.
- Popper.js: Biblioteca utilizada por Bootstrap para manejar popovers y tooltips.
- Bootstrap JS: Incluye componentes dinámicos de Bootstrap como modales y menús desplegables.

20. Añadir_productos.php

PHP: Verificar sesión y rol

```
<?php
// Iniciar la sesión
session_start();

// Verificar si el usuario es administrador
if (!isset($_SESSION['usuario']) || $_SESSION['rol'] != 'administrador') {
    header("Location: login.php");
    exit();
}
```

Iniciar la sesión (`session_start()`):

- Permite usar las variables de sesión para verificar si el usuario está autenticado.

Verificación de permisos:

- Comprueba que el usuario haya iniciado sesión (`isset($_SESSION['usuario'])`) y que su rol sea `administrador`.
- Si no cumple con estas condiciones, redirige al archivo `login.php` y detiene la ejecución con `exit()`.

Procesar el formulario (Agregar un producto)

```php
// Verificar si el formulario ha sido enviado
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Incluir el archivo de conexión a la base de datos
    include('db.php');

    // Obtener los valores del formulario y sanitizarlos
    $nombre = mysqli_real_escape_string($conn, $_POST['nombre']);
    $descripcion = mysqli_real_escape_string($conn, $_POST['descripcion']);
    $precio = mysqli_real_escape_string($conn, $_POST['precio']);
    $stock = mysqli_real_escape_string($conn, $_POST['stock']);

    // Insertar el nuevo producto en la base de datos
    $sql = "INSERT INTO productos (nombre, descripcion, precio, stock)
            VALUES ('$nombre', '$descripcion', '$precio', '$stock')";

    // Verificar si la inserción fue exitosa
    if (mysqli_query($conn, $sql)) {
        $mensaje = "<div class='alert alert-success text-center mt-3'>Producto añadido exitosamente.</div>";
    } else {
        $mensaje = "<div class='alert alert-danger text-center mt-3'>Error al añadir producto: " . mysqli_er
    }

    // Cerrar la conexión a la base de datos
    mysqli_close($conn);
}
?>
```

Validar la solicitud del formulario:

- Comprueba que el formulario fue enviado con el método `POST`.

Incluir conexión a la base de datos:

- Usa `include('db.php')` para conectar con la base de datos.

Sanitizar los datos del formulario:

- Limpia los valores de entrada con `mysqli_real_escape_string` para evitar inyecciones SQL.
- Los datos procesados son:
    - `nombre`: Nombre del producto.
    - `descripcion`: Descripción detallada del producto.
    - `precio`: Precio unitario del producto.
    - `stock`: Cantidad disponible del producto.

Insertar producto:

- Usa una consulta `INSERT` para añadir el nuevo producto en la tabla `productos`.
- Si el producto es añadido correctamente, se almacena un mensaje de éxito.
- En caso de error, almacena un mensaje de error indicando el problema.

Cerrar conexión:

- Libera recursos y cierra la conexión con la base de datos.

Encabezado HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Añadir Producto</title>
    <!-- Enlace a Bootstrap CSS para estilo responsivo -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            padding-top: 50px;
            background: url('https://cdn.pixabay.com/photo/2019/03/29/04/35/tools-4088531_960_720.jpg') no-r
            background-size: cover;
            background-blend-mode: overlay;
            background-color: rgba(255, 255, 255, 0.5); /* Ajusta la transparencia aquí */
        }
    </style>
</head>
```

Meta configuración:

- El idioma del documento se establece como español (`lang="es"`).
- Usa la codificación de caracteres UTF-8 para soportar caracteres especiales en español.

Enlace a Bootstrap:

- Incluye el archivo CSS de Bootstrap para facilitar el diseño responsivo.

Estilo personalizado:

- Agrega un fondo con opacidad ajustable y espaciado superior de `50px`.

Mostrar mensajes dinámicos

```php
// Mostrar mensaje de éxito o error
if (isset($mensaje)) {
    echo $mensaje;
}
```

Si se ha procesado el formulario (es decir, la variable `$mensaje` está definida), muestra un mensaje dinámico:

- **Éxito:** Producto añadido correctamente.
- **Error:** Describe el problema ocurrido durante el proceso.

Formulario HTML para añadir productos

```html
<form action="añadir_producto.php" method="post" class="mt-4">
    <div class="form-group">
        <!-- Campo para ingresar el nombre del producto -->
        <label for="nombre">Nombre del Producto:</label>
        <input type="text" id="nombre" name="nombre" class="form-control" required>
    </div>
    <div class="form-group">
        <!-- Campo para ingresar la descripción del producto -->
        <label for="descripcion">Descripción:</label>
        <textarea id="descripcion" name="descripcion" class="form-control" rows="3" required></texta
    </div>
    <div class="form-group">
        <!-- Campo para ingresar el precio del producto -->
        <label for="precio">Precio:</label>
        <input type="number" id="precio" name="precio" step="0.01" class="form-control" required>
    </div>
    <div class="form-group">
        <!-- Campo para ingresar el stock del producto -->
        <label for="stock">Stock:</label>
        <input type="number" id="stock" name="stock" class="form-control" required>
    </div>
    <button type="submit" class="btn btn-primary btn-block">Añadir Producto</button>
    <a href="admin_menu.php" class="btn btn-secondary btn-block">Volver al Menú</a>
</form>
```

Explicación:

1. **Estructura del formulario:**
   - Envía los datos ingresados al mismo archivo `añadir_producto.php` usando el método `POST`.
2. **Campos del formulario:**
   - `nombre`: Texto para el nombre del producto.
   - `descripcion`: Área de texto para una descripción detallada.
   - `precio`: Entrada numérica para el precio (con decimales).
   - `stock`: Entrada numérica para la cantidad disponible.
3. **Botones del formulario:**
   - "Añadir Producto": Envía los datos para procesarlos en el servidor.
   - "Volver al Menú": Redirige al administrador al menú principal.