INDICE

Proyecto Videoclub¶

En los siguientes ejercicios vamos a simular un pequeño proyecto de un Videoclub mediante un desarrollo incremental**.**

IMPORTANTE, Cada clase debe ir en un archivo php separado y dentro de la carpeta *app*.:

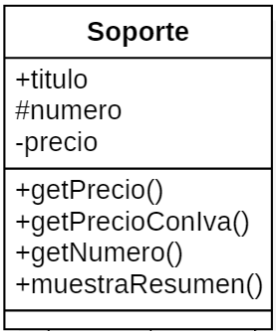
PARTE1 (10.3)

Clases a crear:

Soporte.php ->clase para almacenar los diferentes soportes con los que trabaja nuestro videoclub (cintas de vídeo, videojuegos, etc...). Esta clase será la ***clase padre*** de los diferentes soportes con los que trabaje nuestro videoclub (cintas de vídeo, videojuegos, etc...):

> Crea el constructor que inicialice sus propiedades. Fíjate que la clase no tiene métodos setters.
>
> Definir una constante mediante un propiedad privada y estática denominada `IVA con un valor del 21%

Crear un archivo (inicio.php) para usar las clases y copia el siguiente fragmento:



CÓDIGO DE PRUEBA

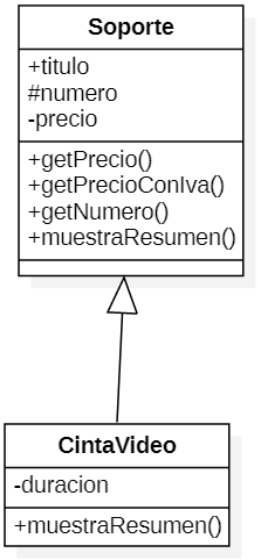
```
<?php
include "Soporte.php";

$soporte1 = new Soporte("Tenet", 22, 3);
echo "<strong>" . $soporte1->titulo . "</strong>";
echo "<br>Precio: " . $soporte1->getPrecio() . " euros";
echo "<br>Precio IVA incluido: " . $soporte1->getPrecioConIVA() . "
euros";
$soporte1->muestraResumen();
```

RESULTADO NAVEGADOR

Tenet
Precio: 3 euros
Precio IVA incluido: 3.63 euros
Tenet
3 € (IVA no incluido)

CintaVideo.php→ la clase CintaVideo *hereda de Soporte*. Añade el atributo ***duracion*** y **sobreescribe** tanto el ***constructor*** como el método ***muestraResumen*** (desde CintaVideo deberás llamar al método muestraResumen del padre).

Añade a inicio.php el código para probar la clase:



CÓDIGO DE PRUEBA

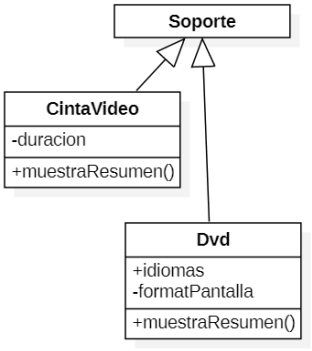
```
<?php
include "CintaVideo.php";

$miCinta = new CintaVideo("Los cazafantasmas", 23, 3.5, 107);
echo "<strong>" . $miCinta->titulo . "</strong>";
echo "<br>Precio: " . $miCinta->getPrecio() . " euros";
echo "<br>Precio IVA incluido: " . $miCinta->getPrecioConIva() . "
euros";
$miCinta->muestraResumen();
```

RESULTADO NAVEGADOR

Los cazafantasmas
Precio: 3.5 euros
Precio IVA incluido: 4.24 euros
Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos

Dvd.php→ la clase Dvd *hereda de Soporte*. Añade los atributos **idiomas** y **formatoPantalla**. A continuación **sobreescribe** tanto el **constructor** como el método **muestraResumen**.

Añade a inicio.php el código para probar la clase:



CÓDIGO DE PRUEBA

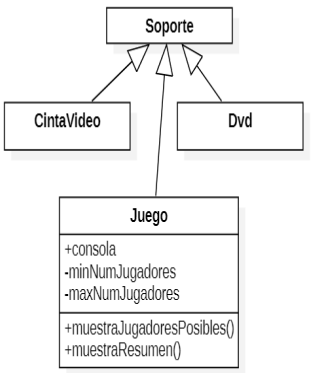
```php
<?php
include "Dvd.php";

$miDvd = new Dvd("Origen", 24, 15, "es,en,fr", "16:9");
echo "<strong>" . $miDvd->titulo . "</strong>";
echo "<br>Precio: " . $miDvd->getPrecio() . " euros";
echo "<br>Precio IVA incluido: " . $miDvd->getPrecioConIva() . " euros";
$miDvd->muestraResumen();
```

RESULTADO NAVEGADOR

Origen
Precio: 15 euros
Precio IVA incluido: 18.15 euros
Origen
15 € (IVA no incluido)
Idiomas: es,en,fr
Formato Pantalla: 16:9

Juego.php→la clase *hereda de Soporte.* Añade los atributos **consola**, **minNumJugadores** y **maxNumJugadores**. A continuación añade el método **muestraJugadoresPosibles**, el cual debe mostrar *Para un jugador*, *Para X jugadores* o De *X a Y jugadores* dependiendo de los valores de las atributos creados. Finalmente, **sobreescribe** tanto el **constructo**r como el método **muestraResumen**.

Añade a inicio.php el código para probar la clase:



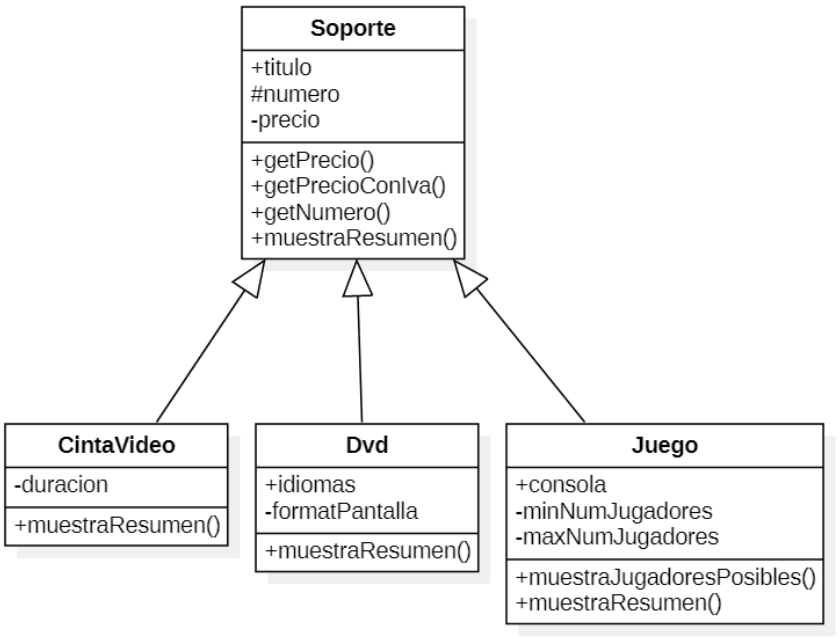
CÓDIGO DE PRUEBA

```php
<?php
include "Juego.php";

$miJuego = new Juego("The Last of Us Part II", 26, 49.99, "PS4", 1, 1);
echo "<strong>" . $miJuego->titulo . "</strong>";
echo "<br>Precio: " . $miJuego->getPrecio() . " euros";
echo "<br>Precio IVA incluido: " . $miJuego->getPrecioConIva() . "
euros";
$miJuego->muestraResumen();
```

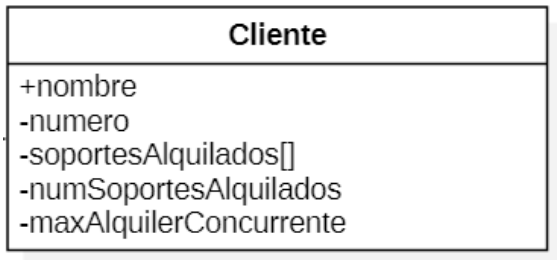
RESULTADO NAVEGADOR

The Last of Us Part II
Precio: 49.99 euros
Precio IVA incluido: 60.49 euros
The Last of Us Part II
49.99 € (IVA no incluido)
Juego para: PS4
Para 1 jugador

En este punto del diseño el modelo debe ser similar al siguiente diagrama:



Cliente.php→. El constructor recibirá el *nombre, numero y maxAlquilerConcurrente*, este último pudiendo ser opcional y tomando como **valor por defecto 3.** Tras ello, añade *getter/setter* únicamente a *numero*, y un *getter* a *numSoportesAlquilados* (este campo va a almacenar un contador del total de alquileres que ha realizado). El array de soportes alquilados contedrá clases que hereden de *Soporte*. Finalmente, añade el método *muestraResumen* que muestre el nombre y la cantidad de alquileres (tamaño del array *soportesAlquilados)*.



Dentro de Cliente, añade las siguiente operaciones:

> ***tieneAlquilado(Soporte $s): bool*** → Recorre el array de soportes y comprueba si está el soporte
>
> ***alquilar(Soporte $s): bool -***→ Debe comprobar si el soporte está alquilado y si no ha superado el cupo de alquileres. Al alquilar, incrementará el *numSoportesAlquilados* y almacenará el soporte en el array. Para cada caso debe mostrar un mensaje informando de lo ocurrido.

devolver(int $numSoporte): bool → Debe comprobar que el soporte estaba alquilado y actualizar la cantidad de soportes alquilados. Para cada caso debe mostrar un mensaje informando de lo ocurrido

listarAlquileres(): void → Informa de cuántos alquileres tiene el cliente y los muestra.

Crea el archivo **inicio2.php** con el siguiente código fuente para probar la clase:

CÓDIGO DE PRUEBA

```
<?php
include_once "CintaVideo.php";
include_once "Dvd.php";
include_once "Juego.php";
include_once "Cliente.php";

//instanciamos un par de objetos cliente
$cliente1 = new Cliente("Bruce Wayne", 23);
$cliente2 = new Cliente("Clark Kent", 33);

//mostramos el número de cada cliente creado
echo "<br>El identificador del cliente 1 es: " . $cliente1->getNumero();
echo "<br>El identificador del cliente 2 es: " . $cliente2->getNumero();

//instancio algunos soportes
$soporte1 = new CintaVideo("Los cazafantasmas", 23, 3.5, 107);
$soporte2 = new Juego("The Last of Us Part II", 26, 49.99, "PS4", 1, 1);
$soporte3 = new Dvd("Origen", 24, 15, "es,en,fr", "16:9");
$soporte4 = new Dvd("El Imperio Contraataca", 4, 3, "es,en","16:9");

//alquilo algunos soportes
$cliente1->alquilar($soporte1);
$cliente1->alquilar($soporte2);
$cliente1->alquilar($soporte3);

//voy a intentar alquilar de nuevo un soporte que ya tiene alquilado
$cliente1->alquilar($soporte1);
//el cliente tiene 3 soportes en alquiler como máximo
//este soporte no lo va a poder alquilar
$cliente1->alquilar($soporte4);
//este soporte no lo tiene alquilado
$cliente1->devolver(4);
//devuelvo un soporte que sí que tiene alquilado
$cliente1->devolver(2);
//alquilo otro soporte
$cliente1->alquilar($soporte4);
//listo los elementos alquilados
$cliente1->listarAlquileres();
//este cliente no tiene alquileres
$cliente2->devolver(2);
```

RESULTADO NAVEGADOR

```
El identificador del cliente 1 es: 23
El identificador del cliente 2 es: 33Alquilado soporte a: Bruce Wayne

Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos
Alquilado soporte a: Bruce Wayne

The Last of Us Part II
49.99 € (IVA no incluido)
Juego para: PS4
Para 1 jugador
Alquilado soporte a: Bruce Wayne

Origen
15 € (IVA no incluido)
Idiomas: es,en,fr
Formato Pantalla: 16:9
El cliente ya tiene alquilado el soporte Los cazafantasmas
Este cliente ya tiene el máximo de alquileres permitidos
No se encontró el soporte en los alquileres
No se encontró el soporte en los alquileres
Este cliente ya tiene el máximo de alquileres permitidos
Alquileres del cliente Bruce Wayne:

Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos

The Last of Us Part II
49.99 € (IVA no incluido)
Juego para: PS4
Para 1 jugador

Origen
15 € (IVA no incluido)
Idiomas: es,en,fr
Formato Pantalla: 16:9
No se encontró el soporte en los alquileres
```

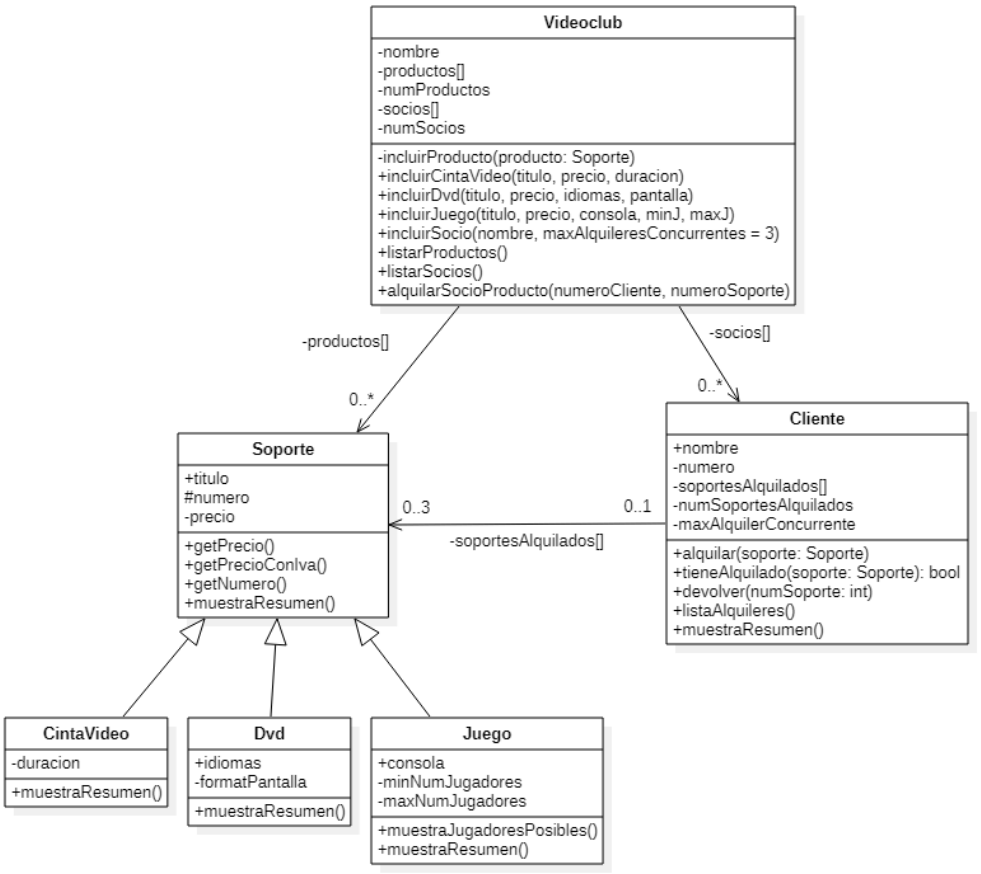
Videoclub.php-> los clientes y los soportes se relacionan mediante esta clase, teniendo en cuenta que:

productos es un array de *Soporte*

socios es una array de *Cliente*

Los métodos públicos de incluir algún soporte, crearán la clase y llamarán al método privado de *incluirProducto*, el cual es el encargado de introducirlo dentro del array.

El modelo completo quedará de la siguiente manera:



Para probar el proyecto, dentro inicio3.php colocaremos:

CÓDIGO DE PRUEBA

```php
<?php
include_once "Videoclub.php"; // No incluimos nada más

$vc = new Videoclub("Severo 8A");

//voy a incluir unos cuantos soportes de prueba
$vc->incluirJuego("God of War", 19.99, "PS4", 1, 1);
$vc->incluirJuego("The Last of Us Part II", 49.99, "PS4", 1, 1);
$vc->incluirDvd("Torrente", 4.5, "es","16:9");
$vc->incluirDvd("Origen", 4.5, "es,en,fr", "16:9");
$vc->incluirDvd("El Imperio Contraataca", 3, "es,en","16:9");
$vc->incluirCintaVideo("Los cazafantasmas", 3.5, 107);
$vc->incluirCintaVideo("El nombre de la Rosa", 1.5, 140);

//listo los productos
$vc->listarProductos();

//voy a crear algunos socios
$vc->incluirSocio("Amancio Ortega");
$vc->incluirSocio("Pablo Picasso", 2);

$vc->alquilaSocioProducto(1,2);
$vc->alquilaSocioProducto(1,3);
//alquilo otra vez el soporte 2 al socio 1.
// no debe dejarme porque ya lo tiene alquilado
$vc->alquilaSocioProducto(1,2);
//alquilo el soporte 6 al socio 1.
//no se puede porque el socio 1 tiene 2 alquileres como máximo
$vc->alquilaSocioProducto(1,6);

//listo los socios
$vc->listarSocios();
```

RESULTADO NAVEGADOR

God of War
19.99 € (IVA no incluido)
Juego para: PS4
Para 1 jugador

The Last of Us Part II
49.99 € (IVA no incluido)
Juego para: PS4
Para 1 jugador

Torrente
4.5 € (IVA no incluido)
Idiomas: es
Formato Pantalla: 16:9

Origen
4.5 € (IVA no incluido)
Idiomas: es,en,fr
Formato Pantalla: 16:9

El Imperio Contraataca
3 € (IVA no incluido)
Idiomas: es,en
Formato Pantalla: 16:9

Los cazafantasmas
3.5 € (IVA no incluido)
Duración: 107 minutos

El nombre de la Rosa
1.5 € (IVA no incluido)
Duración: 140 minutos
Alquilado soporte a: Pablo Picasso

Torrente
4.5 € (IVA no incluido)
Idiomas: es
Formato Pantalla: 16:9
No se pudo alquilar el soporte 2 al cliente 1.
Alquilado soporte a: Pablo Picasso

Origen
4.5 € (IVA no incluido)
Idiomas: es,en,fr
Formato Pantalla: 16:9
No se pudo alquilar el soporte 3 al cliente 1.
El cliente ya tiene alquilado el soporte Torrente
No se pudo alquilar el soporte 2 al cliente 1.
0 - Amancio Ortega
1 - Pablo Picasso

PARTE2 (10.3)

Transforma *Soporte* **a una clase abstracta** y comprueba que todo sigue funcionando. **¿Qué conseguimos al hacerla abstracta?.**

Crea un interfaz *Resumible*, de manera que las clases que lo implementen deben ofrecer el método *muestraResumen().* Modifica la clase *Soporte* y haz que implemente el interfaz. **¿Hace falta que también lo implementen los hijos?**

PARTE3 (10.3)

Modifica las operaciones de ***alquilar***, tanto **en Cliente** como e**n Videoclub**, para dar soporte al encadenamiento de métodos. Posteriormente, *modifica el código de prueba para utilizar esta técnica.*

PARTE4 (10.3)

Haciendo uso de namespaces:

- Coloca todas las clases/interfaces en Dwes\ProyectoVideoclub
- Cada clase debe hacer include_once de los recursos que emplea
- Coloca el/los archivos de prueba en el raíz (sin espacio de nombres)
- Desde el archivo de pruebas, utiliza use para poder realizar accesos sin cualificar

Reorganiza las carpeta tal como hemos visto en los apuntes: **app, test y vendor.**

- Crea un fichero **autoload.php** para registrar la ruta donde encontrar las clases
- **Modifica todo el código**, incluyendo autoload.php **donde sea necesario y borrando los includes previos.**

PARTE5 (10.3)

Crea un conjunto de excepciones de aplicación. Estas excepciones son simples, **no necesitan sobreescribir ningún método**.

Así pues, crea la excepción de la aplicación ***VideoclubException* en** el namespace **Dwes\ProyectoVideoclub\Util.** Posteriormente ***crea los siguientes hijos*** (deben *heredar de VideoclubException*), cada uno en su propio archivo:

SoporteYaAlquiladoException

CupoSuperadoException

SoporteNoEncontradoException

ClienteNoEncontradoException

En ***Cliente***, *modifica* los métodos *alquilar y devolver,* para que hagan **uso de las nuevas excepciones** (lanzándolas cuando sea necesario) **y funcionen como métodos encadenados**. *Destacar que en estos métodos, no se capturan estás excepciones, sólo se lanzan.*

PARTE6 (10.3)

Vamos a modificar el proyecto para que el videoclub sepa qué productos están o no alquilados:

En ***Soporte,*** crea una ***propiedad pública*** cuyo nombre sea ***alquilado*** que *inicialmente* estará *a false*. *Cuando se alquile*, se pondrá *a true*. *Al devolver, la volveremos a poner a false.*

En ***Videoclub***, crea **dos nuevas propiedades y sus getters**:

numProductosAlquilados

numTotalAlquileres

Crea un **nuevo método** en ***Videoclub*** llamado *alquilarSocioProductos(int numSocio, array numerosProductos),* el cual debe recibir un array con los productos a alquilar.

Antes de alquilarlos, debe c**omprobar que todos los soportes estén disponibles**, de manera que si uno no lo está, no se le alquile ninguno.

PARTE7 (10.4)

Para el Videoclub, vamos a crear una página **index.php** con un *formulario que contenga un formulario de login/password.* Se comprobarán los datos en login.php. Los posibles usuarios son admin/admin o usuario/usuario

Si el usuario es correcto, en **main.php** mostrará un mensaje de bienvenida con el nombre del usuari*o, junto a un enlace para cerrar la sesión*, que lo llevaría de nuevo al login.

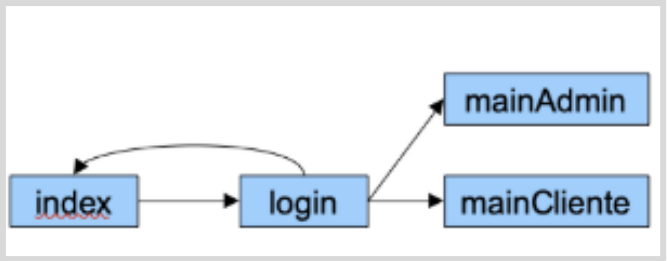
Si el usuario es *incorrecto*, debe *volver a cargar el formulario* dando información al usuario de acceso incorrecto.

A. Si el usuario es administrador (mainAdmin.php), se cargarán en la sesión los datos de soportes y clientes del videoclub que teníamos en nuestras pruebas (ini.php) (*no mediante include sino copiando los datos e insertándolos en un array asociativo, el cual colocaremos posteriormente en la sesión*).

   En mainAdmin.php, además de la bienvenida, debe mostrar: * Listado de clientes * Listado de soportes

   Vamos a **modificar** la clase **Cliente** para almacenar el *user y la password* de cada cliente. Tras codificar los cambios, *modificar el listado de clientes* de *mainAdmin.php* para añadir al listado el usuario.

B. Si el usuario que accede **no es administrado**r *y coincide con alguno de los clientes que tenemos cargados tras el login*, debe cargar ***mainCliente.php*** donde se mostrará un listado de los alquileres del cliente. Para ello, *modificaremos* la clase *Cliente* para ofrecer el método *getAlquileres() : array*, el cual llamaremos y luego recorreremos para mostrar el listado solicitado.



PARTE8 (10.4)

Ofrecer la opción de dar de alta a un nuevo cliente en f*ormCreateCliente.php*.--> Los datos se enviarán mediante POST a *createCliente.php* que *los introducirá en la sesión.* Una vez creado el cliente, debe volver a cargar *mainAdmin.php* donde se podrá ver el cliente insertado. Si hay algún dato incorrecto, debe volver a cargar el formulario de alta.

Crea en *formUpdateCliente.php* un formulario que permita editar los datos de un cliente. Debes recoger los datos en *updateCliente.php* Los datos de cliente se deben poder modificar desde la propia página de un cliente, así como desde el listado del administrador.

Desde el listado de clientes del administrador debes ofrecer la posibilidad de borrar un cliente. En el navegador, antes de redirigir al servidor, el usuario debe confirmar mediante JS que realmente desea eliminar al cliente. Finalmente, en removeCliente.php elimina al cliente de la sesión. Una vez eliminado, debe volver al listado de clientes.

EJ de **listadoClientes.php**

```
<?php
session_start();

// Inicializar la lista de clientes en la sesión si no existe
if (!isset($_SESSION['clientes'])) {
    $_SESSION['clientes'] = ['Cliente1', 'Cliente2', 'Cliente3',
'Cliente4'];
}

// Obtener el listado de clientes
$clientes = $_SESSION['clientes'];
?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Listado de Clientes</title>
```

```
    <script>
        // Confirmación en JavaScript antes de eliminar
        function confirmarEliminacion(cliente) {
            if (confirm(`¿Estás seguro de que deseas eliminar a
${cliente}?`)) {
                // Redirigir al script de eliminación con el cliente
seleccionado
                window.location.href =
`removeCliente.php?cliente=${encodeURIComponent(cliente)}`;
            }
        }
    </script>
</head>
<body>
    <h1>Listado de Clientes</h1>
    <ul>
        <?php foreach ($clientes as $cliente): ?>
            <li>
                <?= htmlspecialchars($cliente) ?>
                <button onclick="confirmarEliminacion('<?=
htmlspecialchars($cliente) ?>')">Eliminar</button>
            </li>
        <?php endforeach; ?>
    </ul>
</body>
</html>
```

ej. **removeCliente.php**

```
<?php
session_start();

// Verificar si se proporcionó un cliente para eliminar
if (isset($_GET['cliente'])) {
    $clienteAEliminar = $_GET['cliente'];

    // Verificar si el cliente está en la lista y eliminarlo
    if (isset($_SESSION['clientes'])) {
        $_SESSION['clientes'] = array_filter($_SESSION['clientes'],
function ($cliente) use ($clienteAEliminar) {
            return $cliente !== $clienteAEliminar;
        });
```

```
    }
}

// Redirigir de vuelta al listado
header("Location: listadoClientes.php");
exit;
```