





DESARROLLO:

Estas trabajando en la actualización de la página web de una tienda de comercio electrónico para permitir a los usuarios buscar y filtrar productos, agregarlos a su carrito de compras, realizar pagos en línea de manera segura y recibir notificaciones sobre promociones y descuentos. Tu próxima tarea consiste en implementar controles mediante el uso de sesiones en PHP dada la posibilidad de riesgo a ser víctimas de fraude en línea, para evitar el robo de información personal y financiera de los clientes, la pérdida de ventas por una brecha de seguridad y el daño a la reputación de la tienda en línea.

Con el fin de incorporar los conocimientos adquiridos en la presente semana, utiliza sesiones en PHP, a través de sus operaciones y funciones, para implementaciones web seguras en la tienda de comercio electrónico.

A continuación, desarrolla las siguientes actividades:

1. Con el fin de proteger la información financiera de los clientes de la tienda de comercio electrónico considerando las operaciones con sesiones en PHP, propone y justifica por lo menos tres medidas específicas para prevenir un ataque cibernético simulado dirigido a robar información.

Medida A — Endurecer la cookie de sesión y el canal de transporte

- Qué hacer:
 - o Forzar HTTPS en todo el sitio y activar HSTS.
 - Configurar la cookie de sesión con: secure=true, httponly=true, samesite='Lax' (o Strict si el flujo lo permite), name personalizado.
- Por qué ayuda: evita el robo de ID de sesión por sniffing (solo viaja por TLS), bloquea acceso
 JavaScript a la cookie (mitiga XSS) y reduce envío cruzado (CSRF por navegación).

Medida B — Prevenir fijación y secuestro de sesión

- Qué hacer:
 - Regenerar el ID de sesión con session_regenerate_id(true) justo al iniciar sesión o al elevar privilegios (checkout, cambios de perfil).
 - Activar session.use_strict_mode=1 para que PHP no acepte IDs inexistentes.
 - Asociar en la sesión "huellas" suaves del cliente (p. ej., hash del agente de usuario) y validar en cada request.
- Por qué ayuda: neutraliza session fixation y dificulta reutilizar sesiones robadas en otro contexto.

Medida C — CSRF tokens para operaciones sensibles

- Qué hacer:
 - Generar token aleatorio y guardarlo en \$_SESSION.
 - Exigirlo y validarlo en formularios/POST críticos (login, agregar tarjeta, checkout).
- Por qué ayuda: evita que un atacante dispare acciones con tu sesión activa desde otro sitio (CSRF).



2. A partir de un escenario o caso de uso relacionado con la experiencia de los clientes con base en criterios de búsqueda personalizados acerca de los productos y servicios ofertados, implementa por lo menos tres medidas para evitar la expiración prematura de las sesiones en la aplicación web de la tienda de comercio electrónico, considerando las diferentes funciones básicas estudiadas.

Medida 1 — Alinear tiempos de cookie y GC

- Qué hacer (servidor y app):
 - o Aumentar session.gc maxlifetime (p. ej., 3600–7200 s según política).
 - Establecer la cookie de sesión con lifetime igual o ligeramente menor a gc maxlifetime.
 - Desacoplar el GC aleatorio (session.gc_probability=0, gc_divisor=1000) y usar un cron/task para limpiar sesiones viejas.
- Por qué ayuda: evita que el recolector borre la sesión mientras el usuario aún navega.

Medida 2 — "Actividad renueva sesión" sin bloquear

- Qué hacer:
 - En cada request real del usuario, actualizar un timestamp en \$_SESSION['last_activity'] y, si corresponde, renovar cookie.
 - Para páginas de listados (mucha lectura), usar session_write_close() tras leer la sesión y así evitar bloqueos que perciben como "cuelgues".
- Por qué ayuda: mantener vivo el estado por uso real, y evitar locks que hacen que el usuario piense que "expiró".

Medida 3 — "Heartbeat" opcional (solo cuando sea necesario)

- Qué hacer:
 - En vistas largas (resultados con scroll/filtrado), una petición AJAX ligera cada X minutos a /keepalive.php que solo refresca la cookie si la pestaña está activa.
- Por qué ayuda: usuarios que comparan por varios minutos no pierden la sesión justo antes de pagar.



3. Con fin de demostrar cómo las sesiones pueden ser utilizadas para almacenar y gestionar los productos seleccionados por los usuarios, escribe el fragmento de código HTML y PHP para desarrollar un prototipo funcional que utilice sesiones para mantener el estado del carrito de compra de productos en el sitio web de la tienda de comercio electrónico.

Inicio de sesión y arreglo del carro PHP

Formulario oculto para enviar productos desde JS a PHP

Modificación de la función addtocart en JS

```
function addToCart(id) {
    const product = products.find(p => p.id === id);
    if(!product) return;

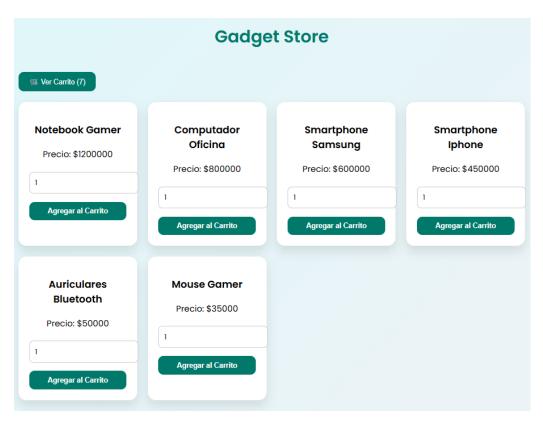
// Colocamos Los valores en el formulario oculto
    document.getElementById('inputProducto').value = product.name;
    document.getElementById('inputPrecio').value = product.price;
    document.getElementById('inputCantidad').value = 1; // puedes cambiar a cantidad seleccionada

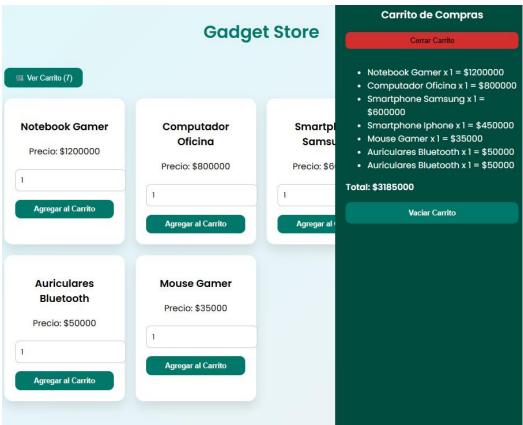
// Enviamos el formulario para que PHP Lo agregue a la sesión
    document.getElementById('formCarrito').submit();
}
```

Visualización del carro desde la sesión



Versión 1

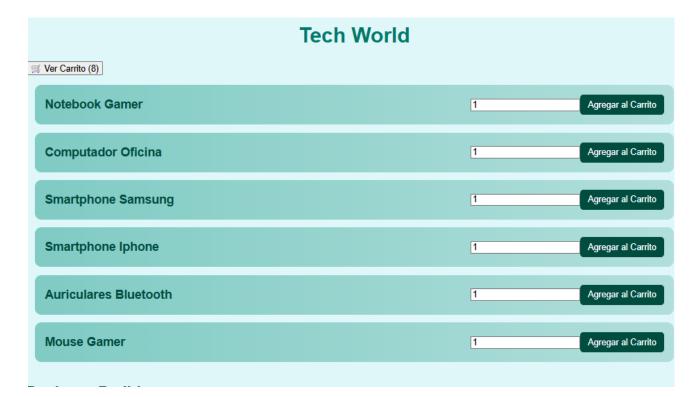




5



Versión 2



Carrito	Tech World	
Cerrar		
 Notebook Gamer x 1 = \$1200000 Computador Oficina x 1 = \$800000 Smartphone Samsung x 1 = \$600000 Smartphone Iphone x 1 = \$450000 Auriculares Bluetooth x 1 = \$50000 Mouse Gamer x 1 = \$35000 Smartphone Iphone x 1 = \$450000 Computador Oficina x 1 = \$800000 Total: \$4385000 Vaciar Carrito 		1 Agregar al Carrito
		Agregar al Carrito
		1 Agregar al Carrito
		1 Agregar al Carrito
		Agregar al Carrito
		1 Agregar al Carrito



Versión 3





7



REFERENCIAS BIBLIOGRÁFICAS

IACC. (2021). Programación Web II en la modalidad online. Programación Web II. Semana 5

Youtube (2020) Sesiones en PHP, ejemplo Carrito de Compras con sesiones. https://www.youtube.com/watch?v=yfuuJt103Cs