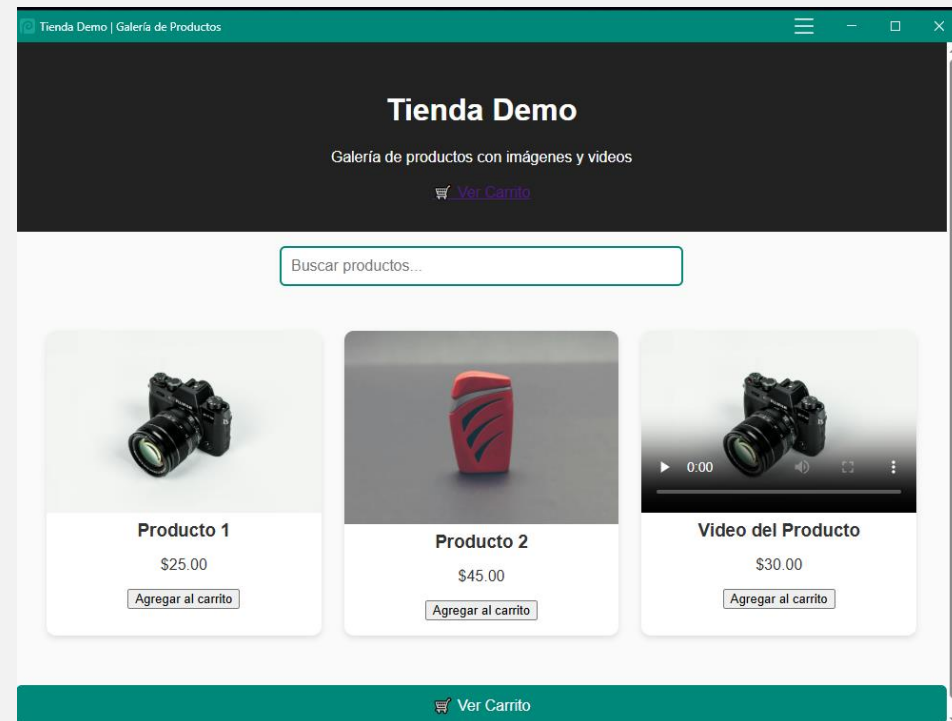
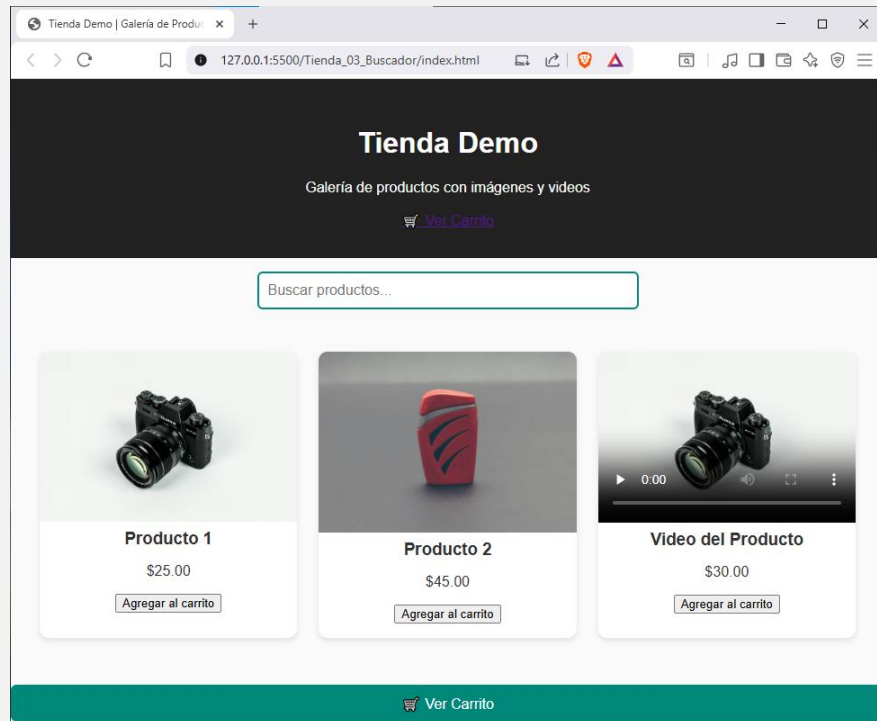




# AEV3 Tienda PWA (Frontend)

2º DAM – Sostenibilidad aplicada al sistema productivo



### Etapas

HTML\_Inicial

PWA\_Inicial

Tienda\_01\_PWA

Tienda\_02\_Carrito

Tienda\_03\_Buscador

### Qué agregas

Solo HTML/CSS básico

manifest + sw.js + registro

Convertir tienda en PWA real

script.js + carrito.html + localStorage

Input + búsqueda en script.js

### Qué aprendes

Estructura de tienda

Cómo funciona una PWA

Offline + instalación

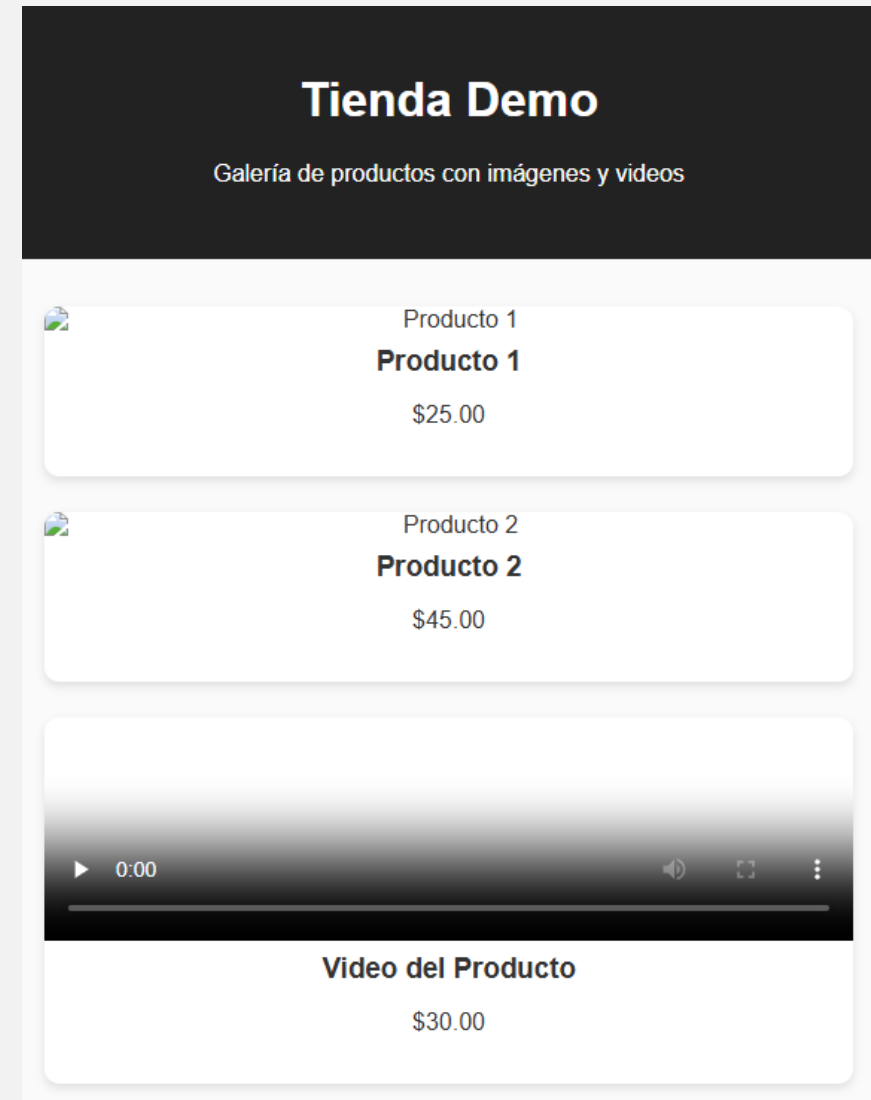
Persistencia

Interactividad avanzada



- `index.html`: una **galería estática** de productos (imágenes/vídeos, títulos, precios...).
- `style.css`: estilos básicos de la tienda (layout, tarjetas de producto, cabecera...).
- `script.js`: prácticamente vacío, solo un `console.log("Galería cargada")`.

👉 Es una **web normal**, sin PWA, sin carrito, sin buscador. Tu objetivo aquí es entender la estructura HTML y los estilos, porque todo lo demás se va a ir construyendo encima de esto.





Objetivo: **aprender los ingredientes mínimos de una PWA.**

1. Crear un archivo **manifest.json** con:

1. nombre, short\_name
2. start\_url
3. display (standalone)
4. Iconos

2. Agregar a **index.html**:

1. `<link rel="manifest" href="manifest.json">`
2. `<meta name="theme-color">`

3. Crear un archivo **sw.js** con:

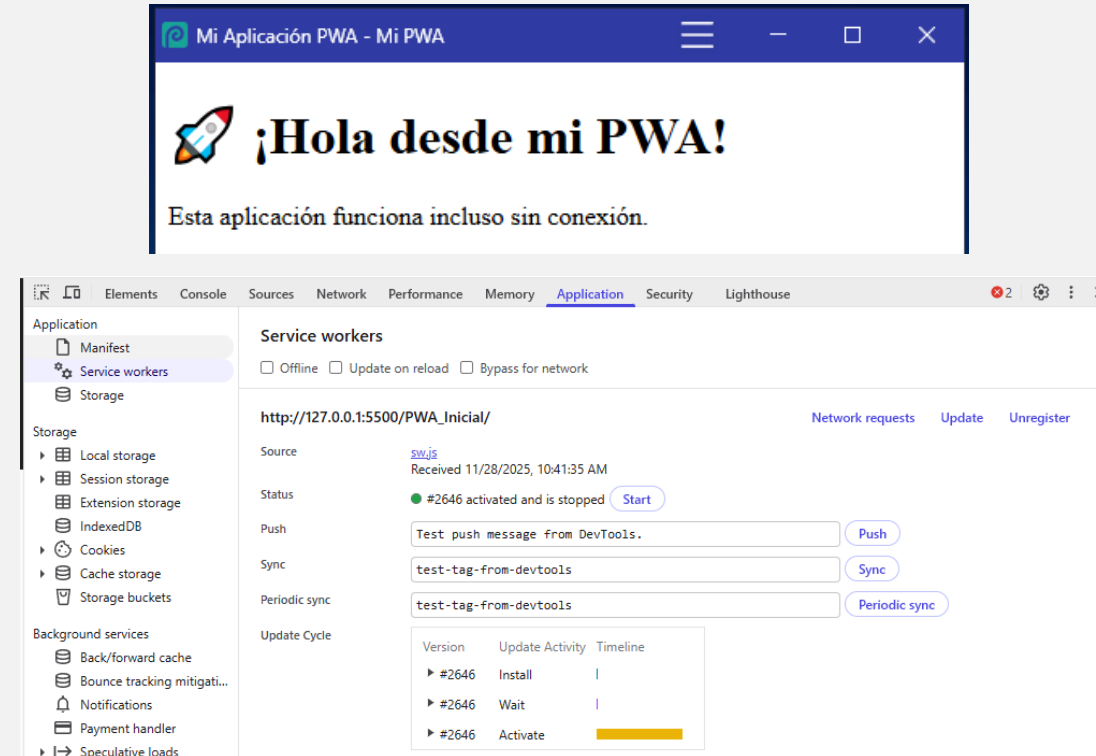
1. Un array de archivos a cachear
2. Evento `install` para cachear esos archivos
3. Evento `fetch` para servir desde cache o red

4. Registrar el service worker en `index.html`.

## ✓ Resultado

Una **PWA muy simple** que puede:

- instalarse en el móvil o escritorio
- funcionar offline mostrando al menos una página



```
PWA_Inicial > JS sw.js > ...
1  const CACHE_NAME = "mi-pwa-v1";
2  const urlsToCache = ["index.html", "manifest.json", "icons/icon-192.png"];
3
4  self.addEventListener("install", (event) => {
5    event.waitUntil(
6      caches.open(CACHE_NAME).then((cache) => cache.addAll(urlsToCache))
7    );
8  });
9
10 self.addEventListener("fetch", (event) => {
11   event.respondWith(
12     caches
13       .match(event.request)
14       .then((response) => response || fetch(event.request))
15   );
16 });
17
```



Objetivo: **convertir la tienda HTML en una PWA real.**

1. Copiar el HTML y CSS de **HTML\_Inicial** como base.

2. Añadir:

1. `<link rel="manifest">`
2. `<meta name="theme-color">`
3. registro de `sw.js`

3. Ajustar `manifest.json` para que represente la tienda:

1. name: "Mi Tienda PWA"
2. icons de tienda

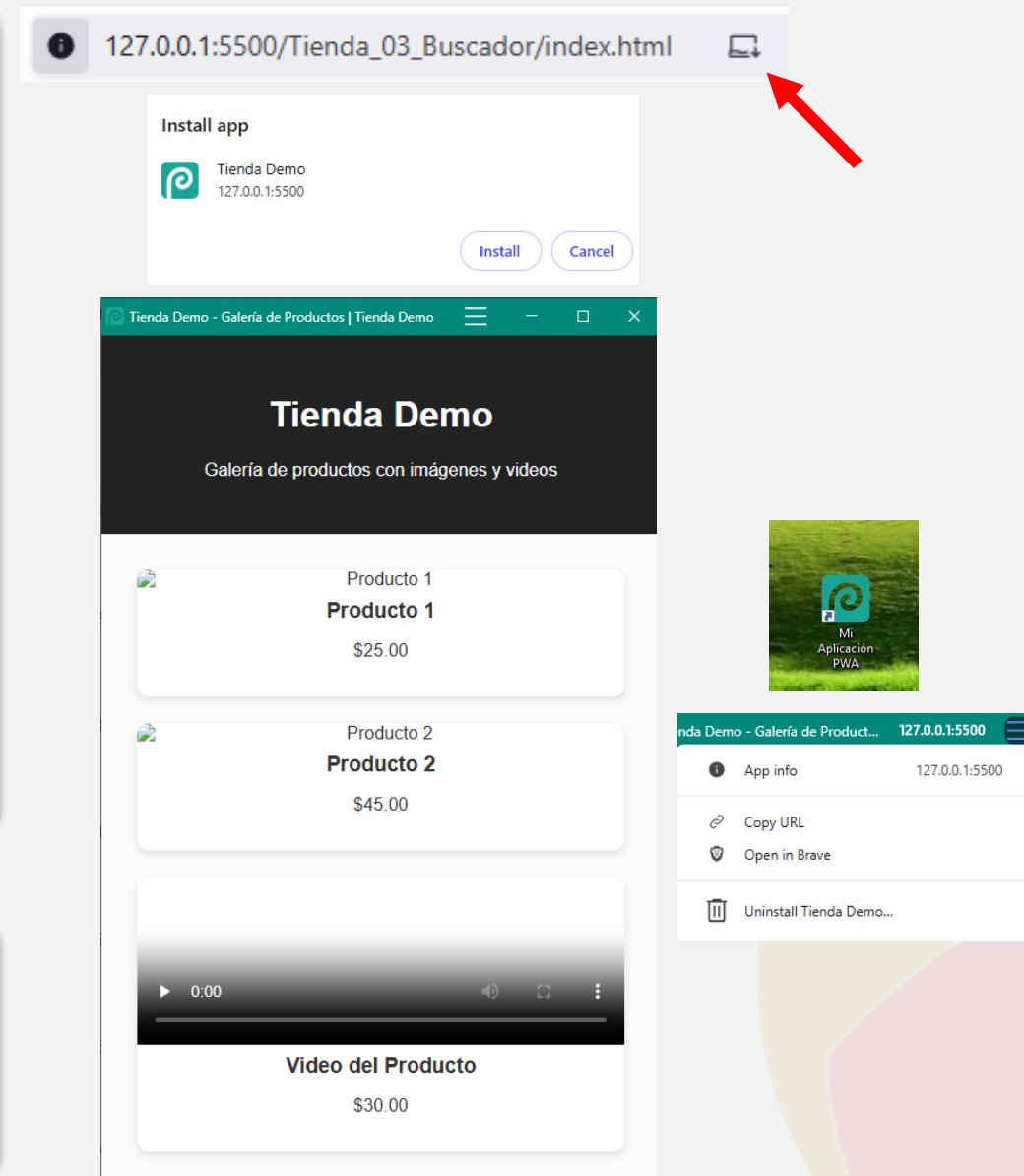
4. Incluir en `sw.js` estos archivos:

1. index.html
2. style.css
3. manifest.json
4. imágenes o logos



### Resultado

Tu tienda ahora es una **PWA instalable**, con la galería de productos funcionando offline (según caché).





Objetivo: **añadir un carrito persistente con localStorage.**

1. Añadir un archivo `script.js`.
2. Crear funciones:
  1. `addToCart(name, price, image)`
  2. Guardar los productos en `localStorage`
  3. Leer el carrito desde `localStorage`
3. Crear `carrito.html` que:
  1. muestra los productos guardados
  2. calcula el total
  3. incluye botón "Vaciar carrito"
4. En `index.html`, añadir botón a cada producto:

```
<button onclick="addToCart('Nombre', 9.99, 'ruta.jpg')">Añadir al carrito</button>
```

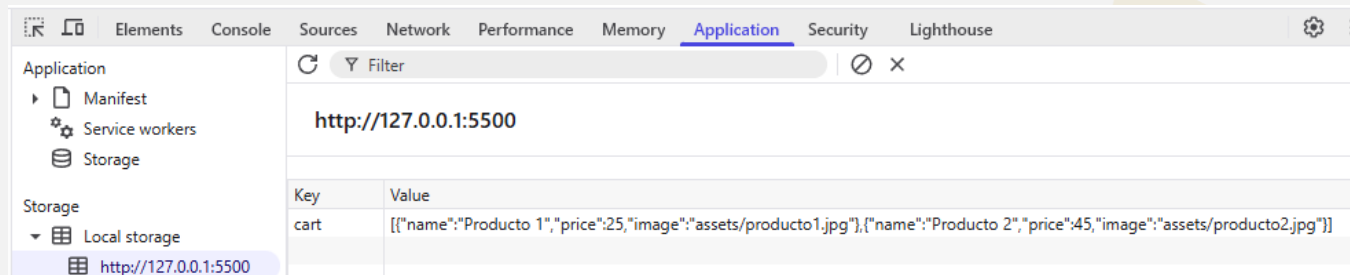
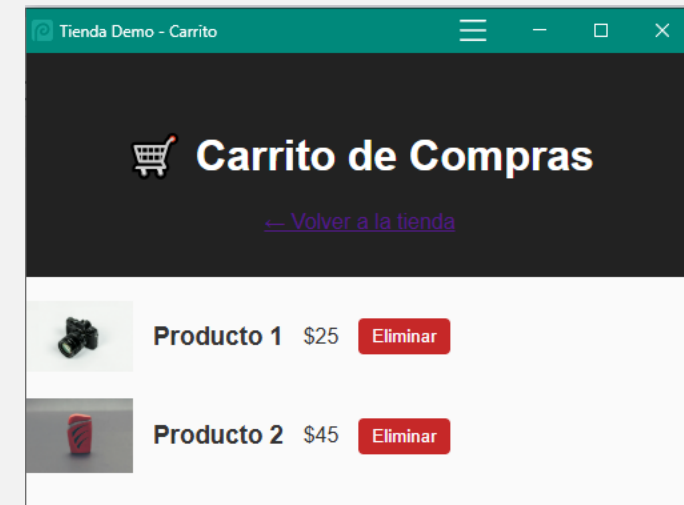
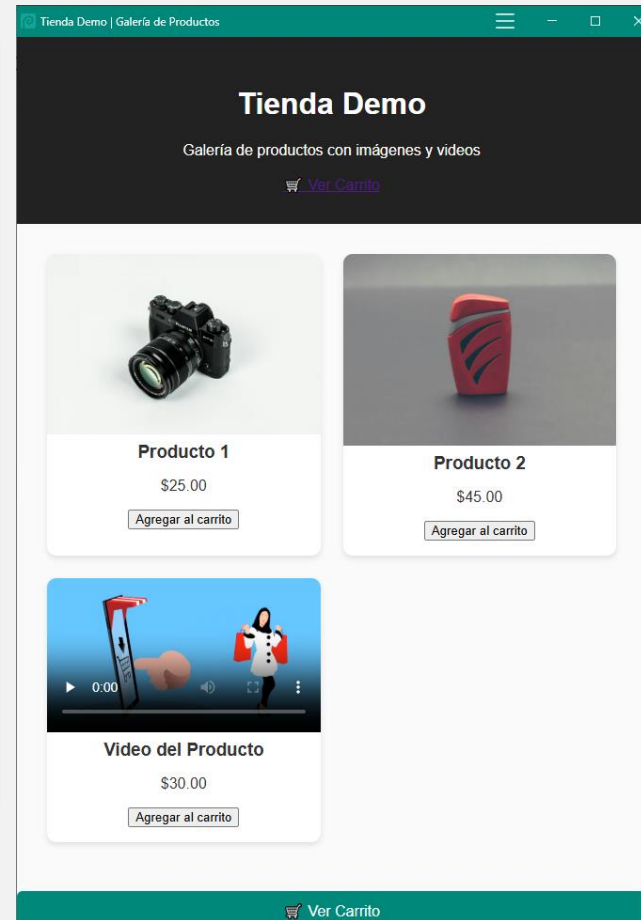
5. Actualizar `sw.js` para cachear:

1. `carrito.html`
2. `script.js`
3. imágenes si quieres que el carrito opere offline

## ✓ Resultado

Tu tienda:

- añade productos al carrito
- guarda todo en `localStorage`
- puede mostrar el carrito en una página separada
- sigue siendo PWA





Objetivo: **añadir un buscador de productos en tiempo real.**

1. Añadir un `<input>` en `index.html`:

```
<input id="searchInput" type="text" placeholder="Buscar..." oninput="searchProducts()">
```

2. En `script.js`, crear la función:

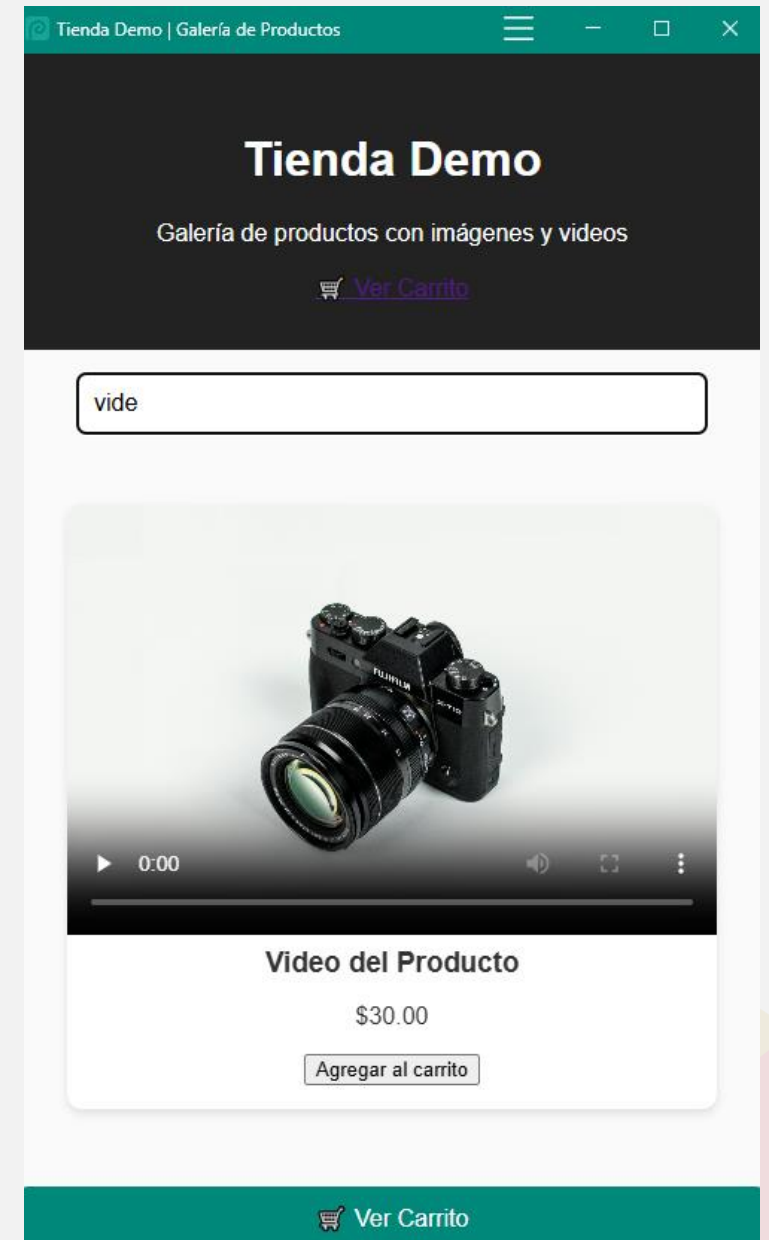
```
function searchProducts() {  
  const text = document.getElementById("searchInput").value.toLowerCase();  
  const products = document.querySelectorAll(".product");  
  
  products.forEach(product => {  
    const name = product.querySelector("h3").textContent.toLowerCase();  
    product.style.display = name.includes(text) ? "block" : "none";  
  });  
}
```

No necesitas modificar manifest ni service worker, solo si agregas nuevos archivos.

### ✓ Resultado

Tu tienda PWA ahora tiene:

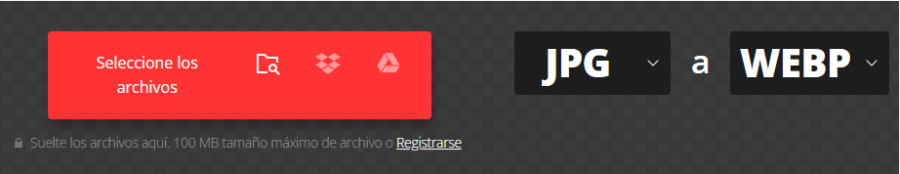
- Búsqueda en tiempo real
- Carrito persistente
- Instalación y funcionamiento offline





Formato	Peso	Calidad	Transparencia	Animación	Compatibilidad	Uso recomendado
WebP	★ Muy ligero	Alta	Sí	Sí	Muy alta (todos los navegadores modernos)	Fotos, gráficos, web en general
AVIF	★ ★ Ultra ligero	★ Alta—muy alta	Sí	Sí	Media—alta	Máxima optimización de peso y calidad
JPEG/JPG	Medio	Alta (fotos)	No	No	Total	Fotografías tradicionales
PNG	Alto	Muy alta (sin pérdida)	Sí	No	Total	Logos, iconos, gráficos con transparencia
SVG	Muy ligero	Perfecta (vector)	Sí	Sí (limitado)	Total	Logos, iconos, ilustraciones escalables
GIF	Alto	Muy baja	Sí (limitado)	Sí	Total	Animaciones simples (pero WebP recomendado)

<https://convertio.co/es/jpg-webp/>







Formato	Peso	Calidad	Códecs comunes	Compatibilidad	Ventajas	Desventajas	Uso recomendado
<b>MP4 (H.264)</b>	Medio	Alta	H.264 / AAC	★ Total	Compatible con todo	No es el más ligero	Videos para cualquier navegador
<b>WebM</b>	★ Ligero	Alta	VP8/VP9/AV1	Muy alta (menos en Safari antiguo)	Archivo más pequeño, moderno	Compatibilidad no absoluta	Web optimizada, videos ligeros
<b>Ogg/Theora (OGV)</b>	Medio	Media	Theora / Vorbis	Media	Software libre	Poco usado, menor calidad	Alternativa abierta, poco común

<https://www.freeconvert.com/es/video-converter>

**Seleccionar Archivos**

Tamaño máximo de archivo 1GB. [Inscribirse](#) Para más información

Al continuar, acepta nuestros [Términos de uso](#).