UNIVERSIDAD PRIVADA DEL VALLE

INGENIERIA DE SISTEMAS INFORMÁTICOS

MANUAL DE USUARIO – SISTEMA WEB TREE

Versión: 1.0

Fecha: 2025

Autor: Equipo de Desarrollo TREE

Documento: Manual de Usuario

Realizado por: Ibañez Flores Mariela Neyza

Asignatura: Programacion Web I

Paralelo: A

Docente: Ing. Miranda Ordoñez Henry

LA Paz - Bolivia

Tabla de contenido

MANUAL DE USUARIO – SISTEMA WEB TREE

1. Introducción y alcance

Este manual extenso describe paso a paso cómo utilizar la tienda web 'TREE' desde la perspectiva de un usuario final, además de explicar el funcionamiento interno del código provisto para administradores y futuros desarrolladores. Incluye guías de uso, flujo de compra, explicación de cada componente del UI, ejemplos de interacción y troubleshooting.

2. Requisitos previos e instalación local

Requisitos mínimos para correr la web localmente:

- Un navegador moderno (Chrome, Edge, Firefox) actualizado.
- Un editor de texto para revisar/editar archivos (VSCode, Sublime, Notepad++).
- Los archivos del proyecto deben estar en la misma carpeta: index.html, SobreNosotros.html, catalogo.html, contacto.html, style.css, app.js y las imágenes referenciadas (logo.jpeg, imagen1.jpg, etc.).
- Para probar en local: abrir index.html directamente en el navegador o servir con un servidor estático (por ejemplo: 'npx http-server' o la extensión Live Server de VSCode).

3. Estructura del proyecto (archivos)

Resumen de archivos y su propósito:

- index.html: Página principal (hero, destacados, newsletter, acceso al catálogo).

- SobreNosotros.html: Información institucional, misión, equipo y preguntas frecuentes.

- catalogo.html: Catálogo dinámico: búsqueda, filtros, orden, paginación y modal de producto.

- contacto.html: Formulario de contacto y mapa integrado (Google Maps).

- style.css: Estilos visuales (variables, layout, responsive, animaciones).

- app.js: Lógica dinámica: inventario, renderizado, modal, carrito, localStorage, eventos.

- imagenes (logo.jpeg, imagen1.jpg, ...): Activos visuales referenciados en HTML/JS.

4. Navegación general y layout

La navegación principal se encuentra en la parte superior (header). Botones principales:
- Inicio: vuelve a index.html
- Sobre Nosotros: información institucional
- Catálogo: listado de productos
- Contactos: formulario y mapa

Elementos globales:
- Botón flotante de carrito (esquina inferior derecha) que abre el panel lateral.
- Menú responsivo: en pantallas pequeñas, el botón hamburguesa alterna la visibilidad del menú.

5. Página: Inicio (index.html) - Descripción y uso

Elementos clave y comportamiento:

- Hero: título, textos persuasivos y botones CTA. 'Ver catálogo' y 'Quiénes somos' llevan a las páginas respectivas.

- Productos destacados: el contenedor #featured se llena desde app.js (función renderFeatured).
- Newsletter: formulario con id newsletterForm. Valida email, guarda en localStorage 'tree_news' y muestra mensaje en #newsletterMsg.

- Flujo de ejemplo (suscripción):
  1) Usuario ingresa su email y presiona 'Suscribirme'.
  2) JS valida y evita duplicados.
  3) Mensaje de confirmación: '¡Gracias! Revisa tu correo.'

6. Página: Sobre Nosotros - Descripción y uso

Contenido: quienés somos, misión, equipo y FAQ. Cada FAQ está implementada con <details> para accesibilidad y una UX compacta. No requiere interacción JS adicional.

7. Página: Catálogo - Funcionalidad detallada

El catálogo es la parte más dinámica. Aquí se documenta por pasos cómo usarlo y qué ocurre internamente.

7.1 Interfaz de usuario (UI)

- Barra de búsqueda (#search): filtra en tiempo real por título o descripción (event listener 'input' en app.js).

- Selector de orden (#sort): llama applySort(mode) al cambiar. Opciones: popular, price-asc, price-desc, new.

- Botones de filtro (Todos, Mujer, Hombre, Accesorios): llaman filterCategory(cat).

- Área de productos (#products): se llena dinámicamente por renderProductsPage(page).

- Paginación (#pagination): renderPagination() crea botones por página según ITEMS_PER_PAGE (6).

7.2 Descripción de comportamiento (paso a paso)

Buscar un producto:

1. En el campo de búsqueda escribe cualquier palabra clave.
2. El listener filtra 'products' por title o desc.
3. workingList se actualiza y se renderiza la primera página de resultados.

Filtrar por categoría:

1. Presiona 'Mujer'/'Hombre'/'Accesorios' o 'Todos'.
2. filterCategory actualiza workingList y llama a renderProductsPage.

Ordenar resultados:

1. Selecciona una opción en el select.
2. applySort ordena workingList (por precio ascendente/descendente o invierte para 'new').

Paginación:

- ITEMS_PER_PAGE = 6. Si hay más productos, renderPagination crea botones numerados. Al presionar un número, se muestran esos items.

Abrir detalles de producto:

- Presiona 'Detalles' en una tarjeta de producto (o en destacados) -> openProductModal(id) construye el contenido del modal y lo muestra.

Añadir al carrito:

- Presiona 'Añadir' en la tarjeta o dentro del modal (añade y cierra modal si procede). La función addToCart(id) gestiona cantidades y llama saveCart().

8. Página: Contacto - Formulario y mapa

Formulario de contacto (id=contactForm):
- Campos: nombre (id=name), teléfono (id=phone), mensaje (id=message).
- Al enviar: se valida nombre y teléfono; se genera una URL para WhatsApp que abre en nueva pestaña con el texto preformateado.
- Uso: útil para conversaciones directas y coordinaciones de venta.

Mapa: iframe embebido con Google Maps. Si necesitas cambiar la ubicación, reemplaza el parámetro 'pb' o utiliza Google Maps -> Compartir -> Insertar mapa -> copia el src.

9. Componentes interactivos

- Modal de producto (#productModal): implementado en HTML y controlado por openProductModal(id) y closeModal().
- Cierre del modal: al click fuera, al presionar ESC o al botón '✕'.
- Panel de carrito (#cartPanel): panel lateral que se muestra/oculta con toggleCart().
- Botón flotante del carrito (.cart-btn): muestra contador total y tiene animación al añadir productos (breatheCartBtn).

10. Carrito de compras - Flujo completo

El carrito tiene persistencia en localStorage (clave 'tree_cart'). A continuación el flujo completo:
- Añadir producto: addToCart(id) -> si existe incrementa qty; si no, lo agrega con qty=1 -> saveCart() -> renderCart() -> breatheCartBtn().
- Ver carrito: toggleCart() muestra panel con renderCart().
- Cambiar cantidades: en el panel, botones '-' y '+' llaman changeQty(id, delta). Si qty llega a 0, se elimina el item.
- Vaciar carrito: clearCart() solicita confirmación antes de borrar.
- Checkout: checkoutWhatsApp() genera el mensaje con el detalle y abre WhatsApp (wa.me) con el texto y total.

Nota para usuario: el checkout abre WhatsApp en el dispositivo; en escritorio requerirá la versión web de WhatsApp o la app instalada.

11. Newsletter y formulario de contacto (comportamiento)

Newsletter:
- El formulario valida que el campo email no esté vacío.
- Se guarda en localStorage en la clave 'tree_news' como un array.
- Evita duplicados y muestra mensajes en #newsletterMsg.

Formulario de contacto:
- Valida nombre y teléfono.
- Prepara texto y abre WhatsApp con la plantilla: 'Hola, soy {name} ({phone})\n\n{message}'.

12. Detalle técnico del script app.js

A continuación se describen las funciones más relevantes, su propósito y extractos de pseudocódigo para entender su lógica. (Si eres desarrollador, estos puntos te ayudarán a mantener o extender el sistema).

products (const)
Array de objetos con campos: id, title, price, cat, img, desc. Representa el inventario embebido en el cliente.

cart (let)
Inicializado desde localStorage: JSON.parse(localStorage.getItem('tree_cart') || '[]').

formatMoney(v)
Devuelve string con prefijo 'Bs. ' y 2 decimales.

renderProductsPage(page)

Calcula start index según ITEMS_PER_PAGE, crea tarjetas HTML y llama renderPagination().

renderPagination()

Calcula total pages = ceil(workingList.length / ITEMS_PER_PAGE) y crea botones numerados; el botón actual queda deshabilitado.

filterCategory(cat)

Si 'all' copia products; si no, filtra por p.cat === cat; reinicia currentPage y renderiza.

applySort(mode)

Soporta 'price-asc', 'price-desc', 'new' (invierte lista). Luego renderProductsPage(1).

openProductModal(id)

Busca producto por id, genera HTML con imagen, título, descripción, precio, botones para añadir y para cerrar; muestra modal y actualiza aria-hidden.

closeModal()

Oculta modal y actualiza aria-hidden.

addToCart(id)

Busca producto; si ya existe incrementa qty; si no lo agrega; llama saveCart() y anima botón del carrito.

saveCart()

Guarda JSON en localStorage y refresca renderCart().

renderCart()

Muestra items con miniaturas, título, precio, controles +/- y el total; actualiza contador en '.cart-btn'.

changeQty(id,delta)

Modifica cantidad; si <=0 remueve el item; guarda carrito.

toggleCart()

Muestra u oculta el panel lateral del carrito y llama renderCart().

clearCart()

Pide confirmación y limpia carrito.

checkoutWhatsApp()

Construye mensaje con detalle (línea por producto) y total; abre wa.me con encodeURIComponent(msg).

renderFeatured()

Toma los 4 primeros productos y genera tarjetas en #featured (usado en index).

13. Detalle técnico del style.css

Resumen de variables globales y reglas destacadas:

- Variables CSS en :root: definen colores, radios, gaps y paleta del sitio. Cambiar estas variables permite recolorear el sitio fácilmente.
- Layout principal utiliza .container con --max-width y .grid para productos (auto-fill / minmax).
- .hero usa display:flex; en pantallas pequeñas se apila (media queries).
- .modal y .cart-panel son posicionados fixed para overlay y panel lateral respectivamente.
- Animaciones: keyframes fadeZoomIn para la imagen del hero y fadeTeam para los miembros del equipo.

14. Accesibilidad y buenas prácticas

Recomendaciones para mejorar accesibilidad:
- Asegurar atributos aria-label en botones importantes (ya presentes en navToggle, mainNav, newsletterForm, contactForm).
- Añadir focus styles visibles (outline) para navegación con teclado.
- Usar texto alternativo descriptivo en todas las imágenes (alt). Verificar que

logo.jpeg y product images tengan alt coherente.
- Revisar contraste de colores (botones sobre imágenes) para cumplir WCAG mínimo 4.5:1 si es necesario.
- Permitir cerrar modal con tecla ESC (ya implementado) y asegurar que el foco se devuelve al elemento que lo abrió.

15. Seguridad, privacidad y manejo de datos

El sitio actualmente almacena datos mínimos en localStorage: carrito (tree_cart) y lista de emails (tree_news). Consideraciones:
- localStorage no es seguro para información sensible. No almacenar datos personales críticos.
- Para producción, manejar pedidos y datos de clientes en servidor seguro (HTTPS + backend) y cumplir normativa local de protección de datos.
- Cuando se usa WhatsApp para checkout, la transferencia de datos se hace por el propio WhatsApp. Informar al usuario antes de enviar datos personales.

16. Resolución de problemas comunes (FAQ técnico-usuario)

- Problema: 'El carrito no guarda entre recargas'.
  Causa y solución:
  - Verificar si el navegador bloquea localStorage (modo incógnito estricto o políticas).
  - Revisar consola del navegador para errores JS (F12).
- Problema: 'El modal no se abre'.
  - Asegurar que los elementos con id productModal y modalContent existan en la página.
  - Revisar consola por errores en openProductModal.
- Problema: 'WhatsApp no abre'.
  - En teléfonos debería abrir la app; en escritorio abrir web.whatsapp si está disponible. Verificar que la URL wa.me esté correcta y no contenga caracteres no permitidos.

17. Personalización y mantenimiento

Cambios frecuentes y dónde realizarlos:

- Cambiar precios o inventario: editar la constante 'products' en app.js. Cada objeto tiene id, title, price, cat, img, desc.
- Añadir o quitar imágenes: subir archivos y actualizar nombres en 'img' del objeto correspondiente.
- Cambiar colores globales: editar :root en style.css (variables como --cta-btn-bg, --catalog-btn-bg).
- Configurar número de WhatsApp: en app.js buscar wa.me/59178539749 y reemplazar el número con el tuyo (formato internacional sin +).
- Modificar ITEMS_PER_PAGE: en app.js cambiar el valor para ajustar cuántos productos se muestran por página.

18. Anexos: fragmentos de código referenciados

- A continuación se incluyen fragmentos clave (copias resumidas del código provisto) con explicación en línea. (Se omiten algunos comentarios para mantener el documento legible).
- Fragmento - Estructura de producto (ejemplo):

```
const products = [{ id:1, title:'Conjunto Mujer', price:200.50, cat:'women', img:'imagen1.jpg', desc:'Talle S - M - L' }, ... ];
```

19. Glosario

- LocalStorage: API del navegador para almacenar pares clave-valor persistentes.

- Modal: Ventana emergente que aparece sobre el contenido principal para mostrar información puntual.

- Responsive: Diseño que se adapta a diferentes tamaños de pantalla.

- CTA: Call To Action, botón que incita a realizar una acción (p.ej. 'Añadir', 'Comprar').

19. Contacto y soporte

Si necesitas ayuda adicional, cambios personalizados o integración con un backend (por ejemplo: pasarela de pagos, panel de administración), describe tus requerimientos y te prepararé una propuesta o el siguiente documento técnico (Manual Técnico o Especificación de Integración).