ERS  
(Tienda Web - “Clan Supply”)

**Asignatura:** DSY1104 Desarrollo Fullstack II

**Proyecto:** Tienda Web “Clan Supply” - Demo

**Integrantes:** Javier Montupil— **Repositorio:** https://github.com/Boychill/TiendaWeb-FullStack-ll

Visión y Alcance

**Objetivo:** entregar un front-end funcional con catálogo, carrito, checkout (demo) y paneles simples de gestión para inventario y pedidos. Persistencia en “LocalStorage”

**Alcance v1:**

* Catálogo con filtros/orden, carrito y checkout que genera pedidos de ejemplo.
* Panel **Gestión de Inventario**: CRUD en tabla + export JSON
* Panel **Gestión de Pedidos**: búsqueda, filtro por estado, edición
* Estilos: categorías como **chips** y **badges** de estado.

Requerimientos

**Requerimientos funcionales**

### **Visitante**

**RF-1 Como Visitante,** quiero ver el catálogo en tarjetas con imagen, nombre y precio, **para** comparar productos rápidamente.

**RF-2 Como Visitante,** quiero buscar productos por nombre, **para** ubicar un artículo específico al instante.

**RF-3 Como Visitante,** quiero filtrar por **categoría** y **subcategoría**, **para** encontrar productos sin revisar toda la lista.

**RF-4 Como Visitante,** quiero ordenar por precio o nombre, **para** priorizar según presupuesto o relevancia.

**RF-5 Como Visitante,** quiero **agregar** productos al **carrito** desde el catálogo, **para** simular la compra sin cambiar de página.

**RF-6 Como Visitante,** quiero ver el **contador del carrito** y el **subtotal/total**, **para** conocer mi gasto acumulado.

**RF-7 Como Visitante,** quiero **editar cantidades** y **eliminar ítems** del carrito, **para** corregir mi selección antes de comprar.

**RF-8 Como Visitante,** quiero **vaciar** el carrito completo, **para** reiniciar mi elección rápidamente.

**RF-9 Como Visitante,** quiero enviar el **formulario de contacto** (con **asunto** y **teléfono** opcional), **para** resolver dudas previas a la compra.

**RF-10 Como Visitante,** quiero ver **chips/etiquetas** de categorías con estados hover/activo, **para** orientarme mejor al navegar.

### **Cliente (registrado)**

**RF-11 Como Cliente,** quiero **iniciar sesión**, **para** mantener mis acciones y preferencias asociadas a mi cuenta.

**RF-12 Como Cliente,** quiero **conservar el carrito** entre recargas, **para** no perder mi avance.

**RF-13 Como Cliente,** quiero **finalizar la compra** (checkout demo), **para** **generar un pedido** en el sistema.

**RF-14 Como Cliente,** quiero recibir un **identificador de pedido** y su **estado**, **para** hacer **seguimiento** de mi compra.

**RF-15 Como Cliente,** quiero ver un **mensaje de confirmación** tras el checkout, **para** saber que la acción se realizó correctamente.

### **Moderador**

**RF-16 Como Moderador,** quiero **listar** y **buscar** pedidos, **para** ubicar rápidamente un pedido específico.

**RF-17 Como Moderador,** quiero **filtrar pedidos por estado**, **para** priorizar el trabajo operativo.

**RF-18 Como Moderador,** quiero **editar el estado**, el **código de seguimiento** y las **notas** del pedido, **para** informar con claridad al cliente.

**RF-19 Como Moderador,** quiero **exportar pedidos a JSON**, **para** respaldar o reportar información.

### **Administrador**

**RF-20 Como Administrador,** quiero **crear** productos con nombre, precio, stock, categoría y subcategoría, **para** mantener actualizado el inventario.

**RF-21 Como Administrador,** quiero **editar** y **eliminar** productos, **para** corregir información o descatalogar.

**RF-22 Como Administrador,** quiero **exportar productos a JSON**, **para** respaldo o migración de datos.

**RF-23 Como Administrador,** quiero **acceder** a un **botón “Admin”** que **abra un modal** con accesos a **Gestión** y **Pedidos**, **para** entrar rápido a las herramientas de administración.

### **Usuario (cualquiera)**

**RF-24 Como Usuario,** quiero navegar con **estructura semántica** (header/nav/main/footer) y enlaces entre páginas, **para** entender la organización del sitio.

**RF-25 Como Usuario,** quiero ver un **perfil** ordenado (avatar, nombre, menú) y poder **cerrar sesión**, **para** gestionar mi cuenta de forma clara.

**Requerimientos No Funcionales**

**RNF-01 — Responsive / Adaptabilidad**

* **Descripción:** El layout se adapta correctamente entre 320px y 1440px.
* **Criterio:** Sin scroll horizontal; grillas se reacomodan en ≥3 breakpoints.
* **Verificación:** Chrome DevTools (iPhone SE, Pixel 7, iPad, 1440px).

**RNF-02 — Accesibilidad (WCAG AA “starter”)**

* **Descripción:** Soporte de lector de pantalla y navegación por teclado.
* **Criterio:**
  + 100% de imágenes con alt.
  + Foco visible en elementos interactivos.
  + label ↔ for en inputs; aria-\* donde aplique.
  + Lighthouse “Accessibility” ≥ **90**.
* **Verificación:** Lighthouse + chequeo visual de foco y etiquetas.

**RNF-03 — Rendimiento de carga**

* **Descripción:** Carga ágil en equipo promedio y servidor local.
* **Criterio:**
  + First Contentful Paint (FCP) < **2.0s** (entorno local).
  + Un solo CSS principal linkeado por vista.
  + Imágenes ≤ **200KB** cada una y con width/height definidos.
* **Verificación:** Lighthouse “Performance” ≥ **85** + inspección de red.

**RNF-04 — Compatibilidad de navegador**

* **Descripción:** Comportamiento consistente.
* **Criterio:** Últimas **2** versiones de Chrome, Firefox y Edge sin errores bloqueantes.
* **Verificación:** Smoke test manual en esos navegadores.

**RNF-05 — Mantenibilidad (código ordenado)**

* **Descripción:** Estructura clara por responsabilidades.
* **Criterio:**
  + CSS separado en estilo.css (sin estilos inline críticos).
  + JS dividido por vista (catálogo, gestión, pedidos).
  + Nomenclatura BEM/simple en clases; sin IDs duplicados.
* **Verificación:** Revisión de estructura y nombres; validador HTML.

**RNF-06 — Calidad de JS (sin malas prácticas)**

* **Descripción:** JS seguro y legible.
* **Criterio:**
  + **0** usos de eval/Function y **0** inyecciones con innerHTML sin sanitizar.
  + Variables/funciones en scopes locales (evitar globales innecesarias).
* **Verificación:** Búsqueda en código + pruebas de formularios.

**RNF-07 — Persistencia local**

* **Descripción:** localStorage para carrito, productos y pedidos demo.
* **Criterio:**
  + Los datos persisten tras recargar.
  + Botones de “limpiar” borran y reflejan UI en < 150 ms.
* **Verificación:** Recarga de página y prueba de limpieza.

**RNF-08 — Estados de interfaz (UX)**

* **Descripción:** Feedback visual coherente.
* **Criterio:**
  + Estados :hover, :focus, :disabled definidos en botones/links/chips.
  + Badges de estado de pedido consistentes (color/leyenda).
* **Verificación:** Inspección visual + hover/focus en elementos clave.

**RNF-09 — Formularios usables**

* **Descripción:** Validaciones claras y no intrusivas.
* **Criterio:**
  + Mensajes de error cerca del campo.
  + Reglas mínimas: email válido; longitudes; teléfono opcional 9–12 dígitos.
  + No hay bloqueo si los campos opcionales van vacíos.
* **Verificación:** Pruebas con entradas válidas/ inválidas.

**RNF-10 — Seguridad (front-only, demo)**

* **Descripción:** Minimizar riesgos típicos en UI.
* **Criterio:**
  + **0** credenciales sensibles hardcodeadas.
  + Sanitizar textos dinámicos antes de insertarlos en el DOM.
  + Namespacing de claves en localStorage (p.ej., app:cart).
* **Verificación:** Revisión de código y llaves de almacenamiento.

**RNF-11 — SEO básico y semántica**

* **Descripción:** Estructura HTML5 clara.
* **Criterio:**
  + Un solo <main> por página; jerarquía h1…h3 correcta.
  + <title> y <meta name="description"> únicos por vista.
* **Verificación:** Validador HTML + inspección de <head>.

**RNF-12 — Mensajería y errores**

* **Descripción:** Errores controlados con mensajes entendibles.
* **Criterio:**
  + **0** errores “uncaught” en consola durante el flujo principal.
  + Mensajes de fallo/éxito visibles en formularios y acciones CRUD.
* **Verificación:** Consola del navegador al recorrer los flujos.

**RNF-13 — Portabilidad / Ejecución local**

* **Descripción:** Fácil de levantar en cualquier equipo.
* **Criterio:**
  + Funciona en servidor estático simple (Live Server o npx serve).
  + **0** rutas rotas relativas (recursos cargan sin file://).
* **Verificación:** Arranque con servidor local y prueba de navegación.

**RNF-14 — Internacionalización (alcance mínimo)**

* **Descripción:** Preparado para textos en español neutro.
* **Criterio:**
  + Textos legibles y sin mezcla innecesaria de idiomas.
  + Separación de strings en un objeto JS si se requiere futura i18n.
* **Verificación:** Revisión de textos y archivo de constantes (si aplica).

**RNF-15 — Documentación**

* **Descripción:** Proyecto entendible para terceros.
* **Criterio:**
  + README.md con cómo correr, estructura y usuarios demo.
  + Sección de **Evidencias** (capturas/GIFs) para la pauta.
* **Verificación:** Presencia y claridad del README + anexos.

Arquitectura de Páginas y Datos

**Home / Catálogo / Carrito**: render dinámico, filtros, ofertas. (main.js)

**Gestión de Inventario**: formulario + tabla editable por fila. (gestion.html + gestion.js)

**Gestión de Pedidos**: lista, búsqueda/filtro, modal de edición. (pedidos.html + pedidos.js)

**Datos (seed)**: productos, usuarios demo, estados de orden. (inventory.js)

Estilos y componente UI

**Categorías estéticas** (.cat-link, .small-chip, hover y active). (estilo.css)

**Badges de estado**: .badge-pendiente, .badge-enviado, etc. (estilo.css)

**Cards de productos** con object-fit y alturas consistentes. (estilo.css)

**Perfil/Auth**: avatar, dropdown y botón “Admin” (cuando rol ≠ user). (main.js)

Casos de Prueba

## **A. Catálogo, Búsqueda y Filtros**

1. **Listado básico** **Pasos:** Abrir index.html.  
    **Resultado:** Se muestra una grilla de tarjetas con imagen, nombre, precio y botón “Agregar”.
2. **Búsqueda parcial (case-insensitive)** **Pasos:** En el buscador, escribir una parte del nombre (p. ej., “cafe”).  
    **Resultado:** Solo aparecen productos cuyo nombre contiene el texto, sin importar mayúsculas.
3. **Sin resultados** **Pasos:** Buscar un término inexistente.  
    **Resultado:** La grilla queda vacía o aparece mensaje de “sin resultados”.
4. **Filtro por categoría** **Pasos:** Click en una categoría (chip/enlace).  
    **Resultado:** Se muestran solo productos de esa categoría; el chip queda activo.
5. **Filtro por subcategoría** **Pasos:** Seleccionar una subcategoría disponible.  
    **Resultado:** Se filtra la lista a esa subcategoría.
6. **Ordenar por precio asc/desc** **Pasos:** Elegir “Precio ↑” y luego “Precio ↓”.  
    **Resultado:** La grilla cambia al orden correspondiente.
7. **Ordenar por nombre** **Pasos:** Elegir “Nombre A→Z”.  
    **Resultado:** Los productos se ordenan alfabéticamente.

## **B. Carrito**

1. **Agregar un producto al carrito** **Pasos:** En la tarjeta, click en “Agregar”.  
    **Resultado:** Aumenta el contador de carrito y el subtotal refleja el precio.
2. **Agregar dos productos distintos** **Pasos:** Agregar dos ítems diferentes.  
    **Resultado:** El contador suma ambos; el total es la suma de subtotales.
3. **Cambiar cantidad** **Pasos:** Abrir panel del carrito, cambiar cantidad de un ítem a 3.  
    **Resultado:** Subtotal del ítem y total general se recalculan.
4. **Eliminar ítem** **Pasos:** Click en eliminar en un ítem.  
    **Resultado:** El ítem desaparece y el total se actualiza.
5. **Vaciar carrito** **Pasos:** Usar la acción “Vaciar carrito”.  
    **Resultado:** Quedan 0 ítems y total en 0.
6. **Persistencia tras recargar** **Pasos:** Con carrito con ítems, recargar la página.  
    **Resultado:** Se mantienen los ítems y el total (persistencia local).
7. **Checkout con carrito vacío (control de flujo)** **Pasos:** Intentar finalizar compra sin ítems.  
    **Resultado:** Se bloquea o muestra mensaje impidiendo avanzar.

## **C. Autenticación y Perfil**

1. **Login correcto (cliente)** **Pasos:** Iniciar sesión con credenciales válidas de usuario cliente.  
    **Resultado:** Aparece el perfil/menú de usuario; carrito y acciones quedan asociadas a la sesión.
2. **Login incorrecto** **Pasos:** Ingresar credenciales inválidas.  
    **Resultado:** Mensaje de error claro; no inicia sesión.
3. **Login admin y acceso rápido** **Pasos:** Iniciar sesión como admin@demo.cl / 123456.  
    **Resultado:** Se muestra botón/entrada “Admin” que abre accesos a Gestión y Pedidos.
4. **Logout** **Pasos:** Cerrar sesión desde el menú de perfil.  
    **Resultado:** Se limpia el estado de usuario; desaparecen accesos de rol.

## **D. Checkout (demo) y Pedidos**

1. **Generar pedido desde carrito** **Pasos:** Logueado, con 2 ítems en el carrito → “Finalizar compra”.  
    **Resultado:** Se crea un pedido con identificador (p. ej., PED-<timestamp>) y redirige a Pedidos.
2. **Pedidos: búsqueda por texto** **Pasos:** En pedidos.html, escribir parte del nombre/código del pedido.  
    **Resultado:** Lista filtrada a los pedidos coincidentes.
3. **Pedidos: filtro por estado** **Pasos:** Seleccionar estado “Pendiente”, luego “Enviado”.  
    **Resultado:** La lista muestra solo pedidos del estado elegido.
4. **Editar estado y guardar** **Pasos:** Abrir modal de edición de un pedido, cambiar a “Entregado”, guardar.  
    **Resultado:** Cambia el badge de estado; el cambio persiste tras recarga.
5. **Agregar tracking y notas** **Pasos:** En el mismo modal, ingresar código de seguimiento y una nota, guardar.  
    **Resultado:** Quedan registrados y visibles al reabrir el pedido.
6. **Exportar pedidos (JSON)** **Pasos:** Click en “Exportar pedidos”.  
    **Resultado:** Se descarga un .json con los pedidos visibles o totales (según implementación).

## **E. Gestión de Inventario**

1. **Crear producto** **Pasos:** Ir a gestion.html, completar nombre, precio, stock, categoría y subcategoría; guardar.  
    **Resultado:** El producto aparece en la tabla y queda persistido.
2. **Editar producto** **Pasos:** Modificar precio y stock de un producto existente; guardar.  
    **Resultado:** La fila se actualiza; el cambio persiste.
3. **Eliminar producto** **Pasos:** Eliminar un producto.  
    **Resultado:** Desaparece de la tabla; no reaparece tras recarga.
4. **Validaciones mínimas** **Pasos:** Intentar guardar con campos obligatorios vacíos o precio/stock inválidos.  
    **Resultado:** Se bloquea el guardado y muestra mensajes de error.
5. **Exportar productos (JSON)** **Pasos:** Click en “Exportar productos”.  
    **Resultado:** Se descarga un .json con el inventario.
6. **Reflejo en catálogo** **Pasos:** Crear/editar un producto y volver a index.html.  
    **Resultado:** El catálogo muestra el cambio (nombre/precio/stock/categoría).

## **F. Formulario de Contacto**

1. **Envío exitoso** **Pasos:** Completar nombre, email válido, mensaje (con asunto/teléfono opcionales); enviar.  
    **Resultado:** Mensaje de confirmación visible; limpia o mantiene el formulario según diseño.
2. **Email inválido** **Pasos:** Poner un email sin “@” o TLD; enviar.  
    **Resultado:** Mensaje de error junto al campo; no se envía.
3. **Teléfono opcional inválido** **Pasos:** Escribir menos de 9 o más de 12 dígitos; enviar.  
    **Resultado:** Mensaje/alerta indicando el formato esperado; no se envía.
4. **Mensaje vacío** **Pasos:** Dejar mensaje en blanco; enviar.  
    **Resultado:** Error “mensaje requerido”; no se envía.

## **G. Navegación y Semántica**

1. **Enlaces de navegación** **Pasos:** Desde index.html, navegar a Gestión, volver a Home y a Pedidos.  
    **Resultado:** Todos los enlaces funcionan; no hay páginas huérfanas.
2. **Semántica por página** **Pasos:** Inspeccionar header, nav, main, footer en cada vista.  
    **Resultado:** Existe un solo <main> y estructura consistente.