DUOC UC - ESCUELA DE INFORMATICA Y TELECOMUNICACIONES

Especificación de Requisitos de Software

Proyecto: Nexbyte Techare

Revisión*: [99.99]*

[Seleccionar fecha]

Especificación de Requisitos según estándar de IEEE 830.

-

Contenido

-

Ficha del documento

Fecha	Revisión	Autor	Modificación

Documento validado por las partes en fecha:

Por el cliente	Por la empresa suministradora
[Firma]	[Firma]
Sr./Sra.	Sr./Sra.

-

1. Introducción

Este documento establece la Especificación de Requisitos de Software para la Versión (V1) de Nexbyte Techcare, filial operativa de Auralis Holdings Spa, con el fin de guiar el desarrollo, la verificación y la evaluación de la solución en el marco de la asignatura.

La ERS alinea tres planos complementarios:

- Académico: requisitos verificables, evidencias y entregables trazables
- Producto: arquitectura front-end puro (HTML5, CSS, JavaScript), validaciones formales y mantenibilidad
- Negocio: rol de Auralis en resguardo de marca, dominios y código, con control de riesgos y enfoque incremental (V1 → V2 con backend), reduciendo dependencia de marketplaces

1.1. Propósito

Establecer la Especificación de Requisitos de Software de la Versión 1 de Nexbyte Techcare Conforme a IEEE-830, como fuente única de verdad que define los requisitos verificables funcionales y no funcionales, sus criterios de aceptación y restricciones del prototipo front-end.

El propósito es guiar el desarrollo, sustentar la demostración evaluable y facilitar la verificación contra la rúbrica de la asignatura, asegurando la trazabilidad con la presentación y el repositorio, y fijando una línea base de alcance para la evolución a V2.

En particular, esta ERS busca:

- Alinear objetivos de negocio Auralis Holdings y Nexbyte Techare con decisiones técnicas como HTML5, CSS, JavaScript, Validaciones, LocalStorage
- Precisar el alcance de V1 y evitar ambigüedades, reduciendo riesgos de sobrepromesa.
- Promover insumos concretos para pruebas de aceptación como lo son casos, evidencias o criterios.
- Servir de contrato técnico y base de control de cambios para iteraciones futuras

1.2. Ámbito del Sistema

Nexbyte Techcare (V1). Proyecto filial de Auralis Holdings SpA, que centraliza PI (marcas, dominios y código) y gobierna los lineamientos de seguridad, identidad y escalamiento. La V1 consiste en front-end puro que entregará un sitio responsive con las páginas Home, Productos, Detalle, Registro, Login, Contacto, Nosotros y un Blog estático. Implementará un carrito con persistencia local en LocalStorage, badge y totales, normalizando cantidades y respetando disponibilidad y/o estado del producto, además de validaciones de formularios en tiempo real al enviar, mostrando mensajes de error junto al campo y bloqueando el envío hasta su corrección. Quedan fuera del alcance de V1 el Backend, APIs y Base de datos, la pasarela de pago, la autenticación real, el CRUD real del módulo de administración, el envío real de correos y el blog dinámico; asimismo, no se almacenarán datos sensibles fuera del navegador.

Con este alcance se busca reducir la dependencia de marketplaces, establecer una base técnica verificable para evolucionar a V2 (con backend y POO) y mantener control de la marca, UX y datos bajo el control de Auralis. Como metas de V1, se contempla el tiempo de carga inicial inferior a dos segundos en entornos de laboratorio, renderizado responsive sin solapes entre 320 y 1440 píxeles, 100% de campos críticos validados en vivo y al enviar mensajes adyacentes y bloqueo del submit, 0 errores en consola durante la demo y compatibilidad estable entre navegadores.

1.3. Definiciones, Acrónimos y Abreviaturas

ERS: Especificación de Requisitos de Software (formato IEEE-830).

V1 / V2: Versión evaluada (front-end puro) / próxima iteración con backend.

Auralis (Auralis Holdings SpA): Matriz que centraliza PI, gobernanza y contratos.

Nexbyte (Nexbyte TechCare): Filial responsable del e-commerce y mantención.

PI: Propiedad Intelectual (marcas, dominios, código fuente y activos asociados).

RUN / DV: Rol Único Nacional y su dígito verificador (0–9/K; cálculo módulo 11).

LocalStorage (LS): Almacenamiento no sensible del navegador para datos de demo.

CRUD: Create, Read, Update, Delete (operaciones básicas de datos).

UI/UX: Interfaz de Usuario / Experiencia de Usuario.

CTA: Llamado a la acción (botón o enlace principal en la interfaz).

1.4. Referencias

Presentación técnica V1 — Nexbyte TechCare (PDF).

Repositorio GitHub — Nexbyte (privado): estructura, commits y README.

DSY1104 — Evaluación Parcial 1 (Instrucciones para Estudiante) — Duoc UC (PDF).

DSY1104 — Experiencia 1 (Rúbrica de evaluación) — Duoc UC (PDF).

DSY1104 — ERS (Plantilla basada en IEEE-830) — Duoc UC (PDF “Tagged”).

MDN Web Docs: Referencias y guías de HTML5, CSS y JavaScript.

Git Documentation (git-scm): Manual oficial de Git (comandos y flujo básico).

GitHub Docs: Flujo de trabajo con repositorios, issues, README y releases.

Conventional Commits — Specification: Estándar de mensajes de commit.

-

Tu Empresa en un Día — Gobierno de Chile: Referencia para constitución de SpA (Auralis).

NIC Chile: Lineamientos de registro de dominios (p. ej., nexbyte.cl).

IEEE 830 (Software Requirements Specification): Estructura recomendada para ERS.

ISO/IEC 25010 (Software product quality): Modelo de calidad (referencial para RNF).

1.5. Visión General del Documento

Este documento presenta la ERS de la V1 de Nexbyte TechCare bajo el paraguas de Auralis Holdings SpA. Su lectura está pensada para combinar claridad académica con foco ejecutivo: primero enmarca el producto y su alcance real; luego fija los requisitos verificables y los criterios de aceptación que permitirán evaluar la entrega; finalmente consolida la información clave para mantener trazabilidad entre lo definido, lo implementado y lo que se demostrará.

La organización es simple y progresiva: comienza con la descripción general del producto, usuarios, restricciones y supuestos; continúa con los requisitos específicos (funcionales con criterios y prioridad, y no funcionales); detalla la arquitectura y datos del front-end; incluye un apartado de trazabilidad; define criterios de aceptación y pruebas; declara entregables y control de versiones; explicita lo fuera de alcance junto al roadmap; y cierra con riesgos y mitigaciones. Como anexo, se incorporan únicamente los formularios de Casos de Uso para complementar el entendimiento de los flujos principales.

2. Descripción General

Nexbyte TechCare (V1) es una aplicación front-end pura construida con HTML5, CSS externo y JavaScript que opera sin backend, base de datos ni servicios remotos. Su objetivo es demostrar de forma verificable la experiencia de compra básica y la disciplina de ingeniería (validaciones, modularidad, accesibilidad y responsive) en un entorno controlado que persiste únicamente datos no sensibles en LocalStorage. Esta primera versión se concibe como línea base tecnológica: una arquitectura simple, mantenible y auditable que habilita la transición ordenada hacia V2 con servicios de servidor.

El sistema se integra en el marco corporativo de Auralis Holdings SpA, que resguarda la propiedad intelectual (marca, dominios y código) y establece lineamientos de gobernanza y seguridad. Desde esta perspectiva, V1 cumple un doble rol: por un lado, materializa una tienda propia que reduce la dependencia de marketplaces y mejora el control de marca/UX/datos; por otro, fija las interfaces y convenciones (estructura de carpetas, contratos de datos en JSON, nombres y eventos de UI) que facilitarán la incorporación futura de un backend sin reescribir la capa de presentación.

La solución está pensada para ejecutarse en navegadores modernos, tanto en escritorio como en dispositivos móviles, con un diseño responsive que garantiza legibilidad y uso fluido en rangos de resolución habituales. Al no depender de APIs externas, la demo es predecible y estable para la evaluación: el contenido de catálogo y el estado de la sesión/carrito se simulan localmente, lo que permite concentrar la validación en comportamientos de interfaz y reglas de negocio que serán extendidas en la siguiente iteración.

2.1. Perspectiva del Producto

Nexbyte TechCare (V1) es un cliente web independiente (HTML5, CSS, JavaScript) que no consume servicios externos ni comparte datos con otros sistemas: ejecuta en el navegador y persiste solo información no sensible mediante LocalStorage. Forma parte del portafolio de Auralis Holdings SpA como activo de PI, pero no tiene dependencias técnicas con la matriz en esta versión.

Aunque V1 es autónoma, el producto está diseñado para integrarse en V2 a un sistema mayor (Catálogo, Usuarios/Auth, Carrito/Pedidos, Pagos, Contenidos y Admin real). Esa integración se prevé vía APIs REST/JSON sobre HTTPS (p. ej., /api/v1/catalog, /api/v1/auth, /api/v1/orders) con contratos versionados y controles estándar (códigos HTTP, tokens). El Admin actual es maqueta: solo ilustra navegación y UI, sin CRUD ni conexión a BD.

Finalmente, V1 se posiciona como standalone verificable en laboratorio; V2 añadirá las interfaces con servicios de plataforma manteniendo la separación cliente/servidor. (Se recomienda acompañar con diagrama de bloques: cliente web ⟷ APIs de plataforma).

2.2. Funciones del Producto

La V1 de Nexbyte TechCare ofrece, en el navegador, la cadena completa de uso: Navegación & Catálogo (Home y listado de productos con imagen/precio/acciones), Detalle (ficha con selección de cantidad válida), Carrito (agregar/editar/eliminar ítems con totales y badge persistente en LocalStorage) y Registro/Login simulados (validaciones RUN+DV, dominios de correo permitidos, teléfono, contraseña con confirmación y región/comuna, con mensajes junto al campo y bloqueo de envío). Se complementa con Contenido informativo (Nosotros y Blog estático) y Contacto (validación y confirmación visual sin envío real). En paralelo, se incluye Administración en modalidad maqueta (dashboard, inventario, reportes, contactos, usuarios) como anticipo del back-office futuro, sin CRUD ni conexión a BD en esta versión.

Flujo funcional resumido: Catálogo → Detalle → Carrito ↔ Registro/Login → Contacto; Admin (maqueta) observa/representa, sin modificar datos en V1.

-

2.3. Características de los Usuarios

La V1 está pensada para dos perfiles reales y acotados al negocio: personas que compran productos tecnológicos y clientes que solicitan mantención de su PC. Ambos interactúan con un sitio web intuitivo y responsive en español, sin procesos complejos ni necesidad de soporte técnico.

Usuario final (Cliente comprador / mantención). Perfil generalista con uso básico de PC y smartphone: navegación web, lectura de fichas de producto, llenado de formularios y acciones simples como “Agregar al carrito” o “Enviar mensaje”. Nivel educacional esperado: enseñanza media completa o formación técnico-profesional inicial. Experiencia: compras online ocasionales y resolución de tareas cotidianas (correo, redes, búsqueda). Competencia técnica: alfabetización digital básica; no requiere conocimientos de hardware o programación. Necesidades clave: catálogo claro (precio y características visibles), flujo Catálogo → Detalle → Carrito sin fricción, validaciones guiadas (RUN, correo, teléfono, región/comuna) y un formulario de contacto para solicitar mantención. La interfaz prioriza textos directos, mensajes de error junto al campo, contraste legible, foco visible y navegación por teclado.

Administrador (gestión interna). Responsable de operación y control básico del e-commerce. Nivel educacional: técnico-profesional; experiencia: conocimientos medios de administración (inventario, orden de información, lectura de indicadores simples) y manejo de ambientes de PC (sistema operativo, carpetas, navegador, nociones de seguridad y Excel/Sheets a nivel medio: filtrar/ordenar/exportar). En V1 su interacción se limita al panel en modalidad maqueta para validar jerarquía de menús, tablas y consistencia visual; en versiones futuras asumirá CRUD real de catálogo, seguimiento de pedidos y atención de solicitudes de mantención.

2.4. Restricciones

Políticas de la empresa. El desarrollo se realiza bajo el paraguas de Auralis Holdings SpA, que centraliza PI, identidad y lineamientos. El repositorio es privado y debe mantener historial claro (Conventional Commits, README y releases); el material público (marcas, dominios, imágenes) sigue guías de marca y uso autorizado.

Limitaciones de hardware/entorno. La V1 está pensada para equipos de laboratorio y dispositivos móviles convencionales. No requiere servidores dedicados ni infraestructura de base de datos; el rendimiento y la memoria dependen del navegador del usuario y de recursos estáticos locales.

Interfaces con otras aplicaciones. No existen integraciones externas en V1: el sistema opera íntegramente en el navegador y persiste datos no sensibles en LocalStorage. No hay consumo de APIs, pasarelas de pago ni servicios de terceros.

Operaciones paralelas. No se soporta concurrencia multiusuario ni procesos en background. La sesión es local y monousuario; el uso simultáneo en múltiples pestañas puede provocar estados divergentes, por lo que la operación recomendada es una pestaña activa.

Funciones de auditoría. No hay logs de servidor ni trazas centralizadas. La trazabilidad se limita al historial Git (commits, tags) y a estados locales del navegador (no aptos como evidencia forense). Cualquier auditoría adicional queda fuera del alcance de V1.

Funciones de control. El módulo de Administración es maqueta: muestra navegación y tablas, pero no ejecuta CRUD real, ni aplica roles, ni restricciones de acceso. La activación/desactivación de elementos se simula a nivel de interfaz.

Lenguajes y dependencias. La solución está restringida a HTML5, CSS externo y JavaScript (vanilla). No se utilizan frameworks ni paquetes de terceros en tiempo de ejecución; el contenido es en español y la estructura de código debe mantenerse modular.

Protocolos de comunicación. V1 sirve archivos estáticos por HTTP/HTTPS (según hosting); no realiza solicitudes de red a APIs. No hay WebSockets ni CORS. Las rutas y recursos deben respetar el mismo origen para evitar errores de carga.

Requisitos de habilidad. El equipo debe dominar fundamentos web (HTML semántico, CSS responsive, JS modular), accesibilidad básica (contraste, foco, teclado) y control de versiones con Git/GitHub. No se exigen habilidades de backend en esta entrega.

Criticidad de la aplicación. La V1 es no crítica para la operación del negocio (prototipo demostrable). No maneja pagos ni datos sensibles; una eventual caída afecta la demo/experiencia, pero no procesos productivos.

Consideraciones de seguridad. No se almacenan datos sensibles fuera del navegador. Se exige validación de entrada en cliente (RUN, correo, teléfono, claves), evitar XSS (escapes básicos), no incrustar scripts de terceros y no incluir secretos en el código. Si se publica, se recomienda HTTPS y cabeceras básicas (CSP, nosniff) en el hosting.

2.5. Suposiciones y Dependencias

Organizacionales. Se asume que Auralis Holdings SpA continúa centralizando PI (marca, dominios y código) y lineamientos de gobernanza, y que Nexbyte TechCare opera como filial responsable del e-commerce y mantención. El repositorio permanece privado y compartido con la docente evaluadora. Cambio esperado: si la estructura corporativa, los permisos del repo o la política de publicación varían (p. ej., apertura del código), podría requerirse ajustar restricciones y trazabilidad definidas en esta ERS.

Tecnológicas. La V1 corre en navegadores modernos (Chrome/Edge/Firefox desktop y móviles actuales) con JavaScript habilitado y soporte de LocalStorage. No hay backend ni APIs; el hosting sirve archivos estáticos por HTTP/HTTPS. Cambio esperado: si se exige compatibilidad con navegadores legacy, ejecución sin JS, o almacenamiento distinto a LocalStorage, deberán revisarse los requisitos funcionales (RF) y no funcionales (RNF) afectados (rendimiento, portabilidad, accesibilidad).

Datos y contenidos. El catálogo y estados se gestionan en LocalStorage y/o JSON local; el blog es estático; no se transmiten datos sensibles. Cambio esperado: si se introduce contenido dinámico, stock en línea o manipulación de datos personales, será necesario incorporar APIs, políticas de datos y nuevos criterios de aceptación (validación, consistencia, auditoría).

Operacionales. Se asume uso monousuario por sesión y una pestaña activa, sin concurrencia ni procesos en background. El módulo Admin es una maqueta (sin CRUD real). Cambio esperado: si se habilita administración efectiva o múltiples sesiones simultáneas, aparecerán requisitos de control de acceso, integridad y estado consistente entre clientes.

Legales y seguridad. No se almacenan datos sensibles fuera del navegador; se aplican validaciones en cliente (RUN/DV, correo con dominios permitidos, teléfono, región/comuna). Cambio esperado: si cambian las políticas de privacidad, dominios de correo admitidos o se integra una pasarela de pago, deberán definirse requisitos de protección de datos, cifrado, registro y cumplimiento (p. ej., términos, consentimientos).

Evolución (dependencia futura V2). El roadmap contempla integrar backend/API/BD, auth real, CRUD de administración, pasarela de pago y blog dinámico mediante APIs REST/JSON. Cambio esperado: cuando esas integraciones se prioricen, esta ERS debe revisarse para incorporar interfaces, contratos, RNF de seguridad/rendimiento y nuevos RF (órdenes, pagos, roles), pudiendo mover límites actuales de “fuera de alcance” a “alcance obligatorio”.

2.6. Requisitos Futuros

A corto/medio plazo (V2) se proyecta la transición desde un front-end autónomo a una plataforma con backend/API/BD que habilite autenticación real, CRUD de administración, checkout con pasarela de pago y blog dinámico. En este escenario, el cliente web consumirá APIs REST/JSON versionadas (catálogo, usuarios, pedidos, pagos, contenido) y se incorporarán roles y permisos para operación interna (administrador, soporte). Para la línea de mantención de PC, se prevé incorporar agendamiento de servicios (solicitud, confirmación, reprogramación), ticketing básico y historial de atenciones por cliente.

En experiencia y conversión, se consideran búsqueda y filtros en catálogo, estado de stock, seguimiento de pedidos, notificaciones (correo/sitio), reseñas moderadas y mejoras de SEO/performance (caché/CDN). En calidad y operación, se agregará auditoría de cambios, métricas (analytics de embudo y panel operativo), exportación a CSV/Sheets, y políticas de respaldo y retención. En arquitectura, se evaluará React u otro framework para componibilidad y pruebas, así como PWA (instalable/offline limitado) y dominio nexbyte.cl bajo Auralis. En seguridad y cumplimiento, se incorporarán protección de datos, cifrado en tránsito/almacenamiento, términos/consentimientos y controles anti-abuso. La incorporación de estas capacidades requerirá una revisión formal de esta ERS,

definiendo contratos de interfaz, nuevos RF/RNF y metas de servicio (p. ej., tiempos de respuesta y disponibilidad).

3. Requisitos Específicos

Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que satisfaga estos requisitos, y que permita al equipo de pruebas planificar y realizar las pruebas que demuestren si el sistema satisface, o no, los requisitos. Todo requisito aquí especificado describirá comportamientos externos del sistema, perceptibles por parte de los usuarios, operadores y otros sistemas. Esta es la sección más larga e importante de la ERS. Deberán aplicarse los siguientes principios:

- El documento debería ser perfectamente legible por personas de muy distintas formaciones e intereses.

- Deberán referenciarse aquellos documentos relevantes que poseen alguna influencia sobre los requisitos.

- Todo requisito deberá ser unívocamente identificable mediante algún código o sistema de numeración adecuado.

- Lo ideal, aunque en la práctica no siempre realizable, es que los requisitos posean las siguientes características:

  - **Corrección**: La ERS es correcta si y sólo si todo requisito que figura aquí (y que será implementado en el sistema) refleja alguna necesidad real. La corrección de la ERS implica que el sistema implementado será el sistema deseado.
  - **No ambiguos**: Cada requisito tiene una sola interpretación. Para eliminar la ambigüedad inherente a los requisitos expresados en lenguaje natural, se deberán utilizar gráficos o notaciones formales. En el caso de utilizar términos que, habitualmente, poseen más de una interpretación, se definirán con precisión en el glosario.
  - **Completos**: Todos los requisitos relevantes han sido incluidos en la ERS. Conviene incluir todas las posibles respuestas del sistema a los datos de entrada, tanto válidos como no válidos.
  - **Consistentes**: Los requisitos no pueden ser contradictorios. Un conjunto de requisitos contradictorio no es implementable.
  - **Clasificados**: Normalmente, no todos los requisitos son igual de importantes. Los requisitos pueden clasificarse por importancia (esenciales, condicionales u opcionales) o por estabilidad (cambios que se espera que afecten al requisito). Esto sirve, ante todo, para no emplear excesivos recursos en implementar requisitos no esenciales.

-

- **Verificables**: La ERS es verificable si y sólo si todos sus requisitos son verificables. Un requisito es verificable (testeable) si existe un proceso finito y no costoso para demostrar que el sistema cumple con el requisito. Un requisito ambiguo no es, en general, verificable. Expresiones como a veces, bien, adecuado, etc. Introducen ambigüedad en los requisitos. Requisitos como “en caso de accidente la nube tóxica no se extenderá más allá de 25Km" no es verificable por el alto costo que conlleva.
- **Modificables**: La ERS es modificable si y sólo si se encuentra estructurada de forma que los cambios a los requisitos pueden realizarse de forma fácil, completa y consistente. La utilización de herramientas automáticas de gestión de requisitos facilitan enormemente esta tarea.
- **Trazables**: La ERS es trazable si se conoce el origen de cada requisito y se facilita la referencia de cada requisito a los componentes del diseño y de la implementación. La trazabilidad hacia atrás indica el origen (documento, persona, etc.) de cada requisito. La trazabilidad hacia delante de un requisito R indica que componentes del sistema son los que realizan el requisito R.

3. Requisitos

A continuación se especifican, con nivel de detalle diseñable y testeable, los comportamientos externos del sistema Nexbyte TechCare — E-commerce (V1). Toda afirmación se formula para ser verificable, no ambigua y trazable con el alcance V1. La numeración sigue el esquema 3.x del índice solicitado.

3.1 Requisitos comunes de las interfaces

(Descripción detallada de entradas/salidas visibles por usuarios y del entorno de ejecución.)

3.1.1 Interfaces de usuario

Modelo de interacción. Aplicación web orientada a navegador, con menú superior fijo (header), área de contenido central y footer. El header contiene: logotipo/branding Nexbyte, enlaces a Home, Productos, Nosotros, Blog, Contacto, accesos Registro/Login y badge de carrito (conteo). El contenido se organiza en tarjetas (cards), formularios con etiquetas visibles sobre el campo y mensajes de error alineados al campo. El footer incluye enlaces informativos y legales.

Estilo visual. Tema oscuro de alto contraste (grises/negros), tipografía sans-serif legible, acentos cromáticos consistentes en botones y enlaces (estado hover/focus/disabled bien diferenciado). Jerarquía tipográfica: h1/h2/h3, textos 14–18px (móvil/desktop), separación vertical generosa. Iconografía coherente (carrito, usuario, contacto).

-

Comportamiento responsive.

≥320 px (móvil): navegación colapsable (menú hamburguesa), grilla 1-columna, imágenes adaptativas sin overflow, sin scroll horizontal.

≥768 px (tablet): grilla 2–3 columnas, header expandido.

≥1024/1440 px (desktop): grilla 3–4 columnas, espacios de respiro; el contenido no excede 1200–1440 px de ancho legible.

Accesibilidad/Usabilidad.

A11y mínima: contraste AA, foco visible en todos los controles, navegación por teclado (tabindex natural), textos alternativos alt en imágenes.

Mensajería de validación: mensajes junto al campo, lenguaje claro (“El RUN no es válido”), no bloquea foco.

Estados de carga/éxito/error: retroalimentación visual inmediata (toasts o banners breves).

Estructura semántica: landmarks (<header> <main> <footer>), títulos jerárquicos, inputs con label for y aria-* cuando aplique.

Entradas de usuario (ejemplos).

Búsqueda simple (UI opcional), selección de categoría (si se maquetó).

Formularios de Registro, Login, Contacto con campos: texto, selectores de Región/Comuna, email, teléfono, contraseña/confirmación.

Controles del Carrito: Agregar, Editar cantidad, Eliminar, Vaciar.

Salidas visibles.

Listado/Detalle con imagen, nombre, precio; totales del Carrito; confirmaciones de acción; errores de validación por campo; estado de sesión simulado (si se usa).

3.1.2 Interfaces de hardware

Dispositivos soportados. PC/laptop con teclado y mouse/trackpad; dispositivos móviles táctiles (teléfonos y tablets). Se requiere: pantalla color, resolución mínima 320×568 px; eventos de puntero/táctil y teclado estándar. No se usa hardware especializado (escáneres, impresoras térmicas, POS, etc.) en V1.

-

Configuración. No requiere instalación. Si se aloja, basta un navegador moderno; si se ejecuta en laboratorio, se abre localmente (archivos estáticos). El viewport móvil se define con meta width=device-width, initial-scale=1.

3.1.3 Interfaces de software

Integraciones externas (V1). No existen. V1 no consume APIs ni SDKs de terceros.

APIs del navegador utilizadas.

DOM para renderizado y eventos.

LocalStorage para estado no sensible: carrito, sesión simulada, usuarios de demo, mensajes.

History/Location para navegación simple.

Timers para feedback UI (toasts).

(No se usan Service Workers, WebSockets ni Fetch para datos de negocio en V1).

Formato de datos (interno).

Producto: { id:string, nombre:string, precio:number, categoria:string, descripcion:string, imagen:string, activo:boolean, stock?:number }

ItemCarrito: { id:string, qty:number }

Usuario (demo): { run:string, nombre:string, apellidos:string, correo:string, pass:string, telefono:string, region:string, comuna:string, direccion:string, createdAt:number }

Auth (simulado): { correo:string, run:string, nombre:string, ts:number }

Mensaje: { id:string, nombre:string, email:string, mensaje:string, createdAt:number }

Claves LocalStorage. nxv3_products, nxv3_cart, nxv3_users, nxv3_auth, nxv3_msgs.

(El prefijo y versión pueden variar; mantener consistencia.)

Compatibilidad. Requiere navegadores con soporte de localStorage y ES6. Si localStorage falla (modo privado estricto), la UI debe degradar con aviso.

3.1.4 Interfaces de comunicación

Transporte. Archivos estáticos servidos por HTTP/HTTPS. No hay comunicaciones con otros sistemas en V1 (sin REST, SOAP, WebSockets ni CORS).

-

V2 (referencial). Se prevé HTTPS + REST/JSON (p. ej., /api/v1/catalog, /api/v1/auth, /api/v1/orders) con autenticación token-based. No forma parte de V1.

3.2 Requisitos funcionales

(Acciones fundamentales ante entradas del usuario, procesamiento y salidas; valida entradas, secuencias, respuestas a situaciones anómalas y datos en cliente.)

Convenciones. ID = RF-NN; Prioridad: MUST/SHOULD/COULD.

Dado–Cuando–Entonces (D-C-E) se usa para claridad de prueba.

Anormalidades: se indican respuestas ante entradas inválidas o estados inconsistentes.

3.2.1 RF-01 — Navegación global (MUST)

Descripción. Header y footer persistentes con accesos a páginas públicas y badge de carrito.

D-C-E.

Dado el usuario en cualquier vista, cuando hace clic en un enlace del header/footer, entonces la app navega sin 404, resalta el enlace activo y preserva el estado visual del carrito.

Dado un usuario que tabula, cuando recorre la navegación, entonces el foco es visible y el orden es lógico.

Anormalidades. Enlace roto → mensaje “Sección no disponible” y retorno seguro a Home.

Salidas. Contenido de la sección objetivo; badge con conteo actualizado.

3.2.2 RF-02 — Home (MUST)

Descripción. Portada con propuesta y CTA “Ver productos”; accesos a Registro/Login.

D-C-E. Dado Home, cuando pulsa “Ver productos”, entonces se abre Productos; en 320–1440 px, sin solapes.

Anormalidades. Recursos gráficos no cargan → se muestra placeholder con alt.

3.2.3 RF-10 — Listado de productos (MUST)

Descripción. Productos lista ≥8 ítems con imagen/nombre/precio y acciones “Ver detalles/Agregar”.

-

D-C-E.

Dado el listado, cuando pulsa Ver detalles, entonces se abre la ficha correcta.

Dado un ítem, cuando pulsa Agregar, entonces se añade 1 unidad al carrito y el badge aumenta.

Parámetros. Filtro/orden (si maquetados) son locales, sin consulta remota.

Anormalidades. Datos corruptos (precio no numérico) → mostrar “No disponible”.

3.2.4 RF-11 — Detalle de producto (MUST)

Descripción. Ficha con info completa y selector de cantidad.

D-C-E.

Dado la ficha, cuando ingresa cantidad <1 o >99, entonces el valor se normaliza a 1 o 99 con aviso.

Dado producto inactivo o sin stock (si aplica), cuando intente agregar, entonces botón deshabilitado + mensaje.

Salidas. Confirmación visual al agregar; retorno opcional a Productos.

3.2.5 RF-20 — Carrito: agregar/editar/eliminar (MUST)

Descripción. Gestionar ítems del carrito con totales en tiempo real.

D-C-E.

Agregar: sumatoria si el producto ya existe (hasta tope).

Editar: actualizar subtotales/total sin recarga.

Eliminar: quitar ítem; si vacío, mostrar estado “Carrito vacío”.

Anormalidades. Cantidad no numérica → normalizar y avisar; item inexistente → ignorar operación.

3.2.6 RF-21 — Carrito: persistencia (MUST)

Descripción. Mantener el carrito en LocalStorage hasta “Vaciar”.

D-C-E. Dado un carrito con datos, cuando recarga o cambia de sección, entonces los ítems y el badge persisten.

-

Anormalidades. localStorage disabled/full → aviso “No se pudo guardar el carrito”; continuar con estado en memoria (sesión).

3.2.7 RF-30 — Registro con validación (MUST)

Descripción. Registrar usuario con validación en vivo y al enviar; campos: nombre, apellidos, RUN + DV, email (dominios permitidos), teléfono, contraseña + confirmación, dirección, región y comuna.

Validaciones.

RUN/DV: 7–8 dígitos + DV 0–9/K (cálculo módulo 11); formatos con/sin puntos/guion normalizados; DV incorrecto rechazado.

Email: solo @duoc.cl, @profesor.duoc.cl, @gmail.com; otros rechazados.

Teléfono: 9 XXXXXXXX (9 dígitos, empieza en 9).

Contraseña: 4–10 caracteres; confirmación idéntica.

Región/Comuna: obligatorias; comunas dependientes de región.

Unicidad: RUN o correo duplicados en base local → rechazo.

D-C-E. Dado campos inválidos, cuando intenta enviar, entonces se bloquea el envío, se muestra error por campo y el foco va al primer error.

Salidas. Confirmación de registro en éxito; limpieza de campos sensibles.

3.2.8 RF-31 — Login simulado (SHOULD)

Descripción. Validar email (dominios permitidos) y contraseña; crear sesión simulada.

D-C-E. Dado credenciales incorrectas, cuando envía, entonces se muestra error general y el foco vuelve al primer campo; en éxito, redirige a Productos o última pública.

Anormalidades. localStorage no disponible → sesión en memoria (se pierde al recargar).

3.2.9 RF-40 — Páginas informativas (SHOULD)

Descripción. Nosotros y Blog estático accesibles y legibles, coherentes con marca.

Criterios. Enlaces internos funcionales; jerarquía de títulos; contraste.

3.2.10 RF-41 — Contacto (SHOULD)

Descripción. Formulario Contacto con nombre, email (dominios permitidos) y mensaje; confirmación visual sin envío a terceros.

D-C-E. Al validar, muestra mensaje de éxito y resetea; errores por campo bloquean envío.

-

3.2.11 RF-50 — Administración (maqueta) (COULD)

Descripción. Vistas de back-office (dashboard, inventario, reportes, contactos, usuarios) sin CRUD real; navega y lista datos de demostración.

Criterios. Menús coherentes, tablas legibles; sin escritura persistente.

3.2.x Requisitos lógicos de información (estado en cliente V1)

Almacenamiento (LocalStorage).

Carrito (nxv3_cart): ItemCarrito[] — persistente hasta “Vaciar”.

Usuarios (nxv3_users): solo para demo local (no sensible).

Auth (nxv3_auth): flag de sesión simulada.

Mensajes (nxv3_msgs): opcional, para listar en Admin (maqueta).

Integridad. Validar tipos y unicidad (RUN/correo); normalizar cantidades y formatos de RUN.

3.3 Requisitos no funcionales

3.3.1 Rendimiento

Carga inicial (SHOULD). En equipos de laboratorio, < 2 s (DOM interactivo) con conexión estándar.

Interacción (SHOULD). 95% de acciones UI (clic en navegación, agregar al carrito) con respuesta visible en < 250 ms.

Peso crítico (SHOULD). Recursos críticos (HTML+CSS+JS principal) ≤ 300 KB minificados (orientativo).

Medición. Verificado con DevTools (Network/Performance).

3.3.2 Seguridad

Datos personales (MUST). No transmitir datos personales a terceros; no almacenar datos sensibles fuera del navegador.

Validación del lado cliente (MUST). Reglas de entrada (RUN, email, teléfono, contraseña) estrictas; sanitización básica para evitar XSS evidente (no inyectar HTML desde inputs).

Secretos (MUST). No embebidos en código (tokens/llaves).

-

Publicación (SHOULD). En hosting, servir por HTTPS y aplicar cabeceras mínimas (CSP básica, X-Content-Type-Options: nosniff, Referrer-Policy).

Sesión simulada (MUST). Debe contener la mínima información (no sensible) y poder limpiarse fácilmente.

3.3.3 Fiabilidad

Errores en consola (MUST). 0 errores no capturados en el flujo principal (Catálogo → Detalle → Carrito → Registro/Login → Contacto).

Degradación (SHOULD). Si localStorage falla, la app informa y opera en memoria (estado no persistente) sin romper navegación.

3.3.4 Disponibilidad

Entorno académico (MUST). La app debe ejecutar offline desde archivos locales en laboratorio.

Alojamiento (SHOULD). Si se publica, objetivo ≥ 99% mensual (estático con CDN); fuera del alcance controlar uptime del proveedor.

3.3.5 Mantenibilidad

Estructura (MUST). Código modular por responsabilidad (ui/layout, productos, detalle, carrito, validaciones, admin).

Estilo (SHOULD). Nombres descriptivos, funciones pequeñas, comentarios concisos.

Versionado (SHOULD). Git con Conventional Commits y README; releases etiquetadas (v1.0.0).

Responsables (MUST). Mantenimiento a cargo del equipo de desarrollo; ajustes visuales y de contenido pueden ser realizados por el administrador en futuras versiones con CMS (V2).

3.3.6 Portabilidad

Navegadores (MUST). Soporte estable en Chrome, Edge y Firefox actuales (desktop) y navegadores móviles modernos.

Dependencias (MUST). Vanilla HTML/CSS/JS; sin frameworks obligatorios en tiempo de ejecución.

SO (SHOULD). Windows/macOS/Linux y Android/iOS (vía navegador).

-

Empaquetado (SHOULD). Archivos estáticos; sin requisitos de servidor de aplicaciones.

3.4 Otros Requisitos

Legal/Branding. Uso de marca, logotipo e imágenes bajo el paraguas de Auralis; no incluir contenidos con derechos no autorizados.

Idioma y localización. Español (Chile); formatos de teléfono y RUN acordes; textos claros y consistentes.

Accesibilidad documental. Toda imagen relevante debe contar con alt; formularios con labels adecuados; evitar dark patterns.

Evidencia de entrega. Preparar 3 capturas (Home, Registro con error/éxito, Carrito con totales) y un extracto de git log --oneline (5–7 commits) para la evaluación.

Roadmap controlado. Cualquier incorporación de backend, pagos o CRUD real exige revisión formal de esta ERS: añadir RF/RNF/Interfaces y actualizar casos de prueba y trazabilidad.

Notas de verificabilidad y trazabilidad

Cada RF/RNF incluye condiciones testeables con fin claro (pasar/fallar).

Las pruebas mínimas sugeridas (ver 3.2) cubren navegación, catálogo, detalle, carrito, validaciones, contacto, responsive, accesibilidad y seguridad básica.

La trazabilidad se mantiene por IDs (RF-, RNF-) y por la correspondencia con vistas/código y evidencias de demo.