| DUOC UC - ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES |
| --- |
| Especificación de Requisitos de Software |
| *Proyecto: TechStore* |
|  |
| **Revisión*: [99.99]*** |
| **[22 de octubre de 2025]** |

| Especificación de Requisitos según estándar de IEEE 830. |
| --- |

**Contenido**

[**Ficha del documento 3**](#_heading=h.t83qhdoe2ype)

[**1. Introducción 4**](#_heading=h.5pgvg01b6ahb)

[1.1.](#_heading=h.i52zgy2uwyqx) Propósito 4

[1.2.](#_heading=h.e1h6tw7tzuf0) Ámbito del Sistema 4

[1.3.](#_heading=h.a7vvyew0olms) Definiciones, Acrónimos y Abreviaturas 4

[1.4.](#_heading=h.s0512ckkg0h2) Referencias 4

[1.5.](#_heading=h.ukhcmbuzdjmw) Visión General del Documento 4

[**2.**](#_heading=h.3js4z0ee8m90) **Descripción General 5**

[2.1.](#_heading=h.ujits1m5onwo) Perspectiva del Producto 5

[2.2.](#_heading=h.j3j2nv8xvtom) Funciones del Producto 5

[2.3.](#_heading=h.ebop6lded76i) Características de los Usuarios 5

[2.4.](#_heading=h.7mtmvk2ha0hn) Restricciones 5

[2.5.](#_heading=h.3p6urj9a780k) Suposiciones y Dependencias 6

[2.6.](#_heading=h.6ex3qy7yzjk1) Requisitos Futuros 6

[**3.**](#_heading=h.9dl2uginqvo7) **Requisitos Específicos 7**

[3.1](#_heading=h.kx6shglc7ni4) Requisitos comunes de las interfaces 8

[*3.1.1*](#_heading=h.69454r6r8vjh) *Interfaces de usuario 8*

[*3.1.2*](#_heading=h.gbnceyfcbv16) *Interfaces de hardware 8*

[*3.1.3*](#_heading=h.av3suua6ptv8) *Interfaces de software 8*

[*3.1.4*](#_heading=h.ij1iwj49doun) *Interfaces de comunicación 8*

[3.2](#_heading=h.lssv9dw449hg) Requisitos funcionales 9

[3.3](#_heading=h.tx9laslrjsih) Requisitos no funcionales 9

[*3.3.1*](#_heading=h.fgrth7cxosb4) *Requisitos de rendimiento 9*

[*3.3.2*](#_heading=h.a91a2b3g3v4p) *Seguridad 10*

[*3.3.3*](#_heading=h.58r6t84t0war) *Fiabilidad 10*

[*3.3.4*](#_heading=h.dqelf1kykl7f) *Disponibilidad 10*

[*3.3.5*](#_heading=h.b2lo2o72xmhr) *Mantenibilidad 10*

[*3.3.6*](#_heading=h.t25fnpyudmnq) *Portabilidad 10*

[3.4](#_heading=h.3welvrlnes94) Otros Requisitos 10

# Ficha del documento

| **Fecha** | **Revisión** | **Autor** | **Modificación** |
| --- | --- | --- | --- |
| *2025-10-22* | *1.0* | *ag mira* | *Creación inicial* |
|  |  |  |  |

Documento validado por las partes en fecha:

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

# 1. Introducción

## 1.1. Propósito

El propósito de este documento ERS es definir, en detalle, los requisitos funcionales y no funcionales del proyecto “TechStore”, una tienda virtual de artículos tecnológicos.

## 1.2. Ámbito del Sistema

El sistema "TechStore" es una tienda virtual que permite a usuarios navegar y comprar productos tecnológicos. Sus principales funciones incluyen: registro/login de usuarios, visualización de catálogo de productos, gestión de carrito de compras, y proceso de checkout. El sistema no incluye procesamiento de pagos real ni integración con ERP externo.

**Beneficios:**

1. Permitir compra de artículos tecnológicos de forma ágil y segura.
2. Práctica de desarrollo con React + TypeScript + testing automatizado.
3. Aprendizaje sobre estado, routing y pruebas unitarias.

**Objetivos:**

1. Crear un sistema web funcional y responsive.
2. Implementar registro/login básicos y roles de usuario/administrador.
3. Permitir compras simuladas y manejo de inventario mock.

## 1.3. Definiciones, Acrónimos y Abreviaturas

1. **ERS:** Especificación de Requisitos de Software.
2. **React:** Biblioteca de UI basada en componentes.
3. **TS:** TypeScript.
4. **Cart:** Carrito de compras.
5. **Admin:** Usuario administrador (gestiona productos).
6. **Checkout:** Proceso de finalizar la compra.

## 1.4. Referencias

En esta subsección se mostrará una lista completa de todos los documentos referenciados en la ERS.

1. Repositorio principal:<https://github.com/ag-mira-duoc/capibara>
2. Plantilla y rúbrica oficial DSY1104
3. Documentación oficial de React, TypeScript, Vite, Jasmine, Karma

## 1.5. Visión General del Documento

Este documento detalla los requisitos del sitio techstore: contexto y usuarios, restricciones, requisitos funcionales y no funcionales, y especificaciones de las interfaces.

# 2. Descripción General

## 2.1. Perspectiva del Producto

TechStore es un producto independiente, funciona como plataforma web. No depende de otros sistemas, pero puede integrarse a una pasarela de pagos en el futuro.

1. Arquitectura: SPA (Single Page Application) sobre React.
2. Diagrama general: Un frontend web con componentes para Navbar, gestión de productos, carrito, usuario y panel de administrador.

## 2.2. Funciones del Producto

Principales funciones organizadas:

1. Visualizar productos (ProductList, ProductCard)
2. Añadir/quitar/actualizar productos en el carrito (Cart, CartItem)
3. Registro y login de usuario
4. Checkout/confirmación de compra
5. Panel administrador para gestión de productos (solo admin)
6. Búsqueda y filtro de productos
7. Responsive para mobile y desktop

## 2.3. Características de los Usuarios

**Tipos de usuario:**

1. **Usuario estándar:** Puede registrarse, iniciar sesión, comprar productos, ver historial de compras.
2. **Administrador:** Acceso al panel de gestión para agregar, editar o eliminar productos.

**Perfil general:**

1. Usuarios con experiencia básica usando navegadores web.
2. Administrador: manejo básico de sistemas y gestión de inventario.

## 2.4. Restricciones

1. Solo frontend con React/TypeScript (sin backend real).
2. Los datos se guardan localmente (mock/localStorage). No hay persistencia en servidor.
3. Compatible con navegadores web modernos y dispositivos móviles.
4. Uso de Jasmine/Karma y/o Vitest para pruebas unitarias.

## 2.5. Suposiciones y Dependencias

1. El sistema será accedido mediante navegadores modernos (Chrome, Firefox, Edge).
2. Para persistencia real se necesitaría integrar un backend en futuras versiones.
3. Las dependencias principales incluyen React, React Router, Bootstrap, TypeScript.

## 2.6. Requisitos Futuros

1. Integración con pasarela de pagos
2. Persistencia real de usuarios/pedidos con base de datos
3. Notificaciones de estado de compra vía email
4. Historial de pedidos y mejoras en panel admin

# 3. Requisitos Específicos

## 3.1 Requisitos comunes de las interfaces

Descripción detallada de todas las entradas y salidas del sistema de software.

### 3.1.1 Interfaces de usuario

1. Uso de Navbar superior y Footer, menú lateral en admin. Colores base: blanco, azul, detalles neutros.
2. Páginas separadas para Home, Productos, Nosotros, Contacto, Carrito, Admin, Login y Registro. Disposición y estilo responsive mediante Bootstrap.
3. Botones con iconografía adecuada (Bootstrap Icons).
4. Formularios validados (ej. email y teléfono en checkout).

### 3.1.2 Interfaces de hardware

1. Compatible con equipos desktop, notebooks y dispositivos móviles con pantalla touch.

### 3.1.3 Interfaces de software

1. No se integra con software externo actualmente.
2. Arquitectura SPA, inicializable localmente vía npm run dev

### 3.1.4 Interfaces de comunicación

1. No integra APIs externas por ahora. Futuras versiones podrían integrarse vía REST o GraphQL.

## 3.2 Requisitos funcionales

1. El usuario puede visualizar el catálogo de productos.
2. El usuario puede registrar una cuenta y iniciar sesión.
3. El usuario puede añadir, eliminar y modificar cantidades de productos en el carrito.
4. El usuario puede completar el proceso de checkout, ingresando sus datos personales y de envío.
5. El sistema valida entradas en los formularios (ej. email correcto, teléfono).
6. El usuario administrador puede agregar, editar y eliminar productos en el panel correspondiente.
7. El sistema muestra mensajes apropiados en estados vacíos (carrito vacío, errores de validación).
8. El sistema permite filtrar y buscar productos por nombre y categoría.
9. Los datos simulados/mocks se cargan desde archivos locales o localStorage.

## 3.3 Requisitos no funcionales

### 3.3.1 Requisitos de rendimiento

1. La aplicación carga en menos de 3 segundos en dispositivos modernos.
2. El 95% de las operaciones dentro de la SPA ocurren en menos de 500 ms

### 3.3.2 Seguridad

1. No almacena contraseñas ni datos sensibles reales.
2. La simulación de usuario/login requiere credenciales comunes mock.
3. El panel de admin sólo es accesible para usuarios autenticados como admin.

### 3.3.3 Fiabilidad

1. El sistema debe ser estable en uso normal, sin cierres inesperados.
2. Pruebas unitarias cubren los componentes críticos; cambios requieren pasar los test.

### 3.3.4 Disponibilidad

1. El sistema está disponible al usuario durante el horario de clases y para pruebas locales 24x7.

### 3.3.5 Mantenibilidad

1. El código está modularizado en componentes reutilizables.
2. La estructura de carpetas facilita agregar nuevos componentes/páginas.

### 3.3.6 Portabilidad

1. Funciona en cualquier sistema operativo que tenga navegador moderno y Node.js.
2. Puede ser desplegado en cualquier servicio de hosting de aplicaciones frontend (Netlify, Vercel, GitHub Pages).

## 3.4 Otros Requisitos

1. Las pruebas unitarias deben estar implementadas y pasar exitosamente (mínimo 10 casos).
2. Diseño Responsivo
3. Accesibilidad básica (etiquetas,contraste adecuado).
4. Codigo fuente versionado en (GitHub).