DUOC UC - ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

Especificación de Requisitos de Software

Proyecto: Óptica Miroo

Integrantes:

Maira Vidal
Lesly Diaz

Docente: Eduardo Baeza

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

En esta sección se proporcionará una introducción a todo el documento de Especificación de Requisitos Software (ERS). Consta de varias subsecciones: propósito, ámbito del sistema, definiciones, referencias y visión general del documento.

1.1. Propósito

La finalidad de este documento es establecer los requisitos funcionales y no funcionales del sistema de tienda en línea de Óptica Miroo. Está destinado a los desarrolladores, los testers, los interesados y el equipo encargado de la administración del proyecto.

1.2. Ámbito del Sistema

Nombre del sistema: Óptica Miroo: tienda virtual

Que hará el sistema: Permitirá la venta en línea de lentes oftálmicos, lentes de contacto, monturas y accesorios, con un catálogo digital, carrito de compras, registro de clientes, blog informativo y panel administrativo.

Que no hará el sistema: No incluirá pasarela de pago integrada en esta fase (se simulará).

Beneficios del sistema: Optimizar procesos internos, mejorar la experiencia del cliente, expandir el canal de ventas y actualizar la imagen digital.

1.3. Definiciones, Acrónimos y Abreviaturas

- ERS: Especificación de Requisitos de Software
- UI: Interfaz de Usuario
- UX: Experiencia de Usuario
- CMS: Content Management System (Sistema de Gestión de Contenidos)

1.4. Referencias

IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications.

Apuntes de la asignatura Ingeniería de Software, DUOC UC, 2025.

Material de clases y presentaciones del docente Eduardo Baeza.

Sitio web de referencia para estructura de ERS: https://www.ieee.org

Ejemplos de tiendas virtuales y catálogos en línea consultados con fines académicos (no comerciales).

1.5. Visión General del Documento

Esta documentación detalla de forma organizada y precisa todos los requerimientos que se necesitan para el desarrollo de la tienda virtual de Óptica Miroo. Se divide en tres secciones principales:

Introducción: Describe el objetivo general del proyecto, su propósito y las metas específicas.

-

Descripción general: Proporciona una perspectiva comprensiva del sistema, sus funciones esenciales y los usuarios que lo emplearán.

Condiciones específicas: Especifica, paso a paso, las características técnicas que debe satisfacer (como la seguridad, el diseño adaptable y la velocidad) y las funcionalidades que debe tener la tienda (por ejemplo el carrito de compras, el catálogo y el panel de control).

2. Descripción General

Esta sección describe el panorama general del sistema de tienda en línea para Óptica Miroo, explicando los elementos que afectan al producto y a sus requisitos sin adentrarse en los pormenores específicos de estos.

2.1. Perspectiva del Producto

La tienda online de Óptica Miroo es un producto autónomo que moderniza y amplía el canal de ventas actual. Aunque está diseñada para permitir integraciones futuras (como sistemas externos de gestión de inventario o pasarelas de pago reales), en su primera versión funcionará como una plataforma web independiente. Se comunicará directamente con los clientes (usuarios finales) y con los empleados internos de la óptica mediante una arquitectura cliente-servidor, basada en React para la interfaz y Spring Boot para la lógica de negocio.

2.2. Funciones del Producto

El sistema debe permitir:

- Presentar un catálogo digital de productos (monturas, lentes oftálmicos, lentes de contacto y accesorios), con ilustraciones, descripciones y precios, obtenidos desde la base de datos MySQL.
- Administrar una cesta de compra en la que los usuarios puedan añadir artículos, observar un resumen y simular el proceso de compra.
- Brindar a los clientes un sistema seguro de registro e inicio de sesión, con validación en el backend y almacenamiento seguro de credenciales.
- Incluir un blog informativo con textos relacionados con la salud ocular, administrado por colaboradores internos.
- Disponer de un panel administrativo para manejar usuarios, productos y roles de acceso, con operaciones CRUD sobre la base de datos.

2.3. Características de los Usuarios

- Clientes finales: Personas con conocimientos básicos en el uso de navegadores. Acceden al catálogo, añaden productos al carrito y completan compras simuladas.

- Administradores: Usuarios internos con acceso a un panel administrativo para agregar, editar o eliminar productos y gestionar roles.
- Colaboradores de contenido: Personal encargado de cargar textos o imágenes en el blog mediante formularios sencillos dentro del sistema.

2.4. Restricciones

- El sistema se desarrollará con React en el frontend y Spring Boot en el backend.
- Los datos se almacenarán en una base de datos MySQL.
- El proceso de compra será una simulación de flujo, sin integración inicial con pasarelas de pago reales.
- El tiempo de desarrollo estará limitado al período académico.
- El diseño debe ser responsivo, adaptable a computadores, tablets y dispositivos móviles.

2.5. Suposiciones y Dependencias

- Se asume que los usuarios tendrán conexión a internet y usarán navegadores actualizados (Chrome, Firefox, Edge, Safari).
- Se asume que los administradores tendrán conocimientos básicos en el uso de formularios web.
- El correcto funcionamiento depende de la compatibilidad con navegadores modernos.
- Se depende de un entorno de prueba con servidores locales para React, Spring Boot y MySQL, antes de pasar a producción.

2.6. Requisitos Futuros

- Integración de pasarelas de pago reales para completar transacciones.
- Implementación de filtros y búsqueda avanzada en el catálogo.
- Generación de estadísticas (productos más vistos, más agregados al carrito) mediante consultas a la base de datos.
- Mejoras en la personalización visual, como temas configurables o modo oscuro.

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.

3.1 Requisitos comunes de las interfaces

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

-

3.1.1 Interfaces de usuario

La interfaz de usuario contará de páginas web ordenadas en un menú superior con secciones principales como inicio, catálogo, carrito de compras, blog informativo, contacto y área de cuenta del usuario. El diseño será sencillo con el objetivo de facilitar la navegación incluso para usuarios con poca experiencia en tecnología. Los formularios de registro, inicio de sesión y contacto contarán con validaciones realizadas en JavaScript para verificar datos básicos como formato de correo electrónico o campos obligatorios. El catálogo mostrará productos acompañados de imágenes, descripciones y precios, cargados de forma estática o a través de almacenamiento local. Finalmente, el diseño será hecho mediante CSS, de modo que la plataforma pueda visualizarse correctamente en pantallas de diferentes tamaños, tanto en computadores como en dispositivos móviles.

3.1.2 Interfaces de hardware

El sistema podrá utilizarse en computadores personales, tablets y teléfonos inteligentes sin necesidad de hardware especializado. Los usuarios podrán interactuar mediante teclado y mouse en computadores de escritorio, así como mediante pantallas táctiles en dispositivos móviles. Gracias a su desarrollo basado en React para la interfaz y Spring Boot para los servicios de backend, la aplicación mantiene un rendimiento ligero y eficiente. No requerirá equipos de alto rendimiento, sino únicamente un dispositivo con un navegador actualizado y acceso a internet para consumir las funcionalidades expuestas por el servidor.

3.1.3 Interfaces de software

La plataforma será compatible con navegadores modernos como Google Chrome, Mozilla Firefox, Microsoft Edge y Safari. No será necesario instalar software adicional en los dispositivos de los usuarios, ya que el sistema se ejecutará directamente desde el navegador con conexión a internet. El almacenamiento de información, como los productos agregados al carrito o los datos de sesión de los usuarios, se gestionará mediante un servidor backend en Spring Boot conectado a una base de datos MySQL, lo que garantiza persistencia real de los datos, seguridad y escalabilidad. De esta forma, la aplicación no dependerá de mecanismos locales como localStorage, sino de un entorno centralizado y confiable para el manejo de la información.

3.1.4 Interfaces de comunicación

El sistema se comunicará de manera interna mediante los componentes y eventos definidos en React, gestionando la navegación a través de un enrutador dinámico que permite transiciones fluidas entre las distintas secciones de la plataforma. A diferencia de la versión anterior, ahora se establecerán conexiones reales con el servidor backend en Spring Boot, el cual expone servicios REST que interactúan con la base de datos MySQL. Esto permitirá realizar acciones como el registro de usuarios, la gestión de sesiones o el envío de correos electrónicos mediante servicios integrados, sin depender de simulaciones locales como mailto:. De esta forma, la aplicación ofrece una experiencia completa y robusta, basada en protocolos de comunicación estándar (HTTP/HTTPS) y en una arquitectura cliente-servidor escalable.

-

3.2 Requisitos funcionales

3.2.1 Catálogo digital

El sistema debe mostrar un catálogo digital con productos de la óptica, incluyendo nombre, imagen, descripción y precio. Los datos se obtendrán desde la base de datos MySQL mediante servicios REST expuestos por el backend en Spring Boot, y se renderizarán dinámicamente en la interfaz desarrollada con React.

3.2.2 Carrito de compras

Los usuarios deben poder añadir productos al carrito de compras, visualizar un resumen de los artículos seleccionados y calcular el valor total estimado. La información del carrito se gestionará en el frontend con React y se sincronizará con el backend para garantizar persistencia y consistencia de datos.

3.2.3 Proceso de compra

El proceso de compra se implementará mediante un formulario que recoja los datos del cliente y los envíe al backend en Spring Boot. El sistema registrará la orden en la base de datos MySQL y mostrará un mensaje de confirmación. Aunque no se realizarán transacciones financieras reales, la arquitectura permitirá integrar pasarelas de pago en el futuro.

3.2.4 Registro de usuarios

El sistema debe permitir el registro de nuevos usuarios a través de un formulario en React. Los datos ingresados se validarán en el frontend y se enviarán al backend, donde se aplicarán validaciones adicionales y se almacenarán de forma segura en la base de datos MySQL (incluyendo cifrado de contraseñas).

3.2.5 Inicio de sesión de usuarios

Los usuarios registrados deben poder iniciar sesión con sus credenciales. La autenticación se gestionará mediante el backend en Spring Boot, utilizando sesiones seguras o tokens JWT. Las credenciales ya no se almacenarán en el navegador, sino en la base de datos MySQL.

3.2.6 Acceso a cuenta

Cada usuario debe poder acceder a su cuenta para actualizar ciertos datos personales. Las modificaciones se enviarán al backend y se reflejarán en la base de datos MySQL, garantizando persistencia y seguridad.

3.2.7 Formulario de contacto

El sistema debe contar con un formulario de contacto que permita a los clientes enviar consultas o comentarios, simulando el envío mediante un enlace de correo (mailto) o un mensaje en pantalla.

-

3.2.8 Blog informativo

El sistema debe incluir un blog informativo con artículos estáticos relacionados con la salud visual, cargados manualmente por los administradores.

3.2.9 Panel administrativo

El panel administrativo debe permitir la gestión de productos y artículos del blog, ofreciendo funciones para agregar, editar o eliminar registros. Estas operaciones se realizarán mediante servicios REST en Spring Boot y se reflejarán en la base de datos MySQL.

3.3 Requisitos no funcionales

3.3.1 Requisitos de rendimiento

El sistema debe cargar la página principal y el catálogo en un tiempo máximo de 2 segundos en conexiones de al menos 10 Mbps.

El 95% de las interacciones del usuario (navegar entre secciones, añadir productos al carrito, mostrar el resumen de compra) deben completarse en menos de 1 segundo.

El sistema debe soportar hasta 20 usuarios concurrentes en el entorno de pruebas sin degradación perceptible del rendimiento.

Las imágenes del catálogo deben estar optimizadas para no superar los 300 KB por archivo, manteniendo calidad suficiente para pantallas de alta resolución.

3.3.2 Seguridad

- El acceso al panel administrativo debe requerir autenticación mediante usuario y contraseña, gestionada por el backend en Spring Boot, con validación en el cliente y servidor.
- Las contraseñas deben almacenarse en la base de datos MySQL utilizando hash seguro (bcrypt o equivalente), evitando exposición directa.
- Se debe implementar validación de datos en formularios tanto en el frontend (React) como en el backend (Spring Boot) para prevenir ataques de inyección SQL y XSS.
- El sistema debe registrar en un archivo de log gestionado por el backend las acciones críticas realizadas en el panel administrativo (creación, edición o eliminación de productos).

3.3.3 Fiabilidad

El sistema debe mantener la integridad de los datos del carrito mientras el usuario navega entre secciones, incluso si se recarga la página, mediante persistencia temporal en el backend.

La pérdida de datos en el carrito no debe superar el 1% de las sesiones durante las pruebas.

-

En caso de error en la carga de un recurso (imagen o script), el sistema debe mostrar un mensaje de advertencia y permitir continuar la navegación.

3.3.4 Disponibilidad

El sistema debe estar disponible para uso el 99% del tiempo durante el periodo de pruebas académicas.

Las interrupciones planificadas (por mantenimiento o actualización) deben comunicarse con al menos 24 horas de antelación a los usuarios de prueba.

3.3.5 Mantenibilidad

El código debe estar modularizado en componentes React para el frontend y en módulos Spring Boot para el backend, facilitando su lectura y actualización.

Las tareas de mantenimiento (actualización de productos, cambios de precios, carga de artículos en el blog) deben poder ser realizadas por un administrador con conocimientos básicos de ofimática y navegación web.

Se debe realizar una revisión semanal de contenido y datos del catálogo durante el periodo de pruebas.

3.3.6 Portabilidad

- El sistema debe ser compatible con los navegadores Google Chrome, Mozilla Firefox, Microsoft Edge y Safari en sus dos últimas versiones estables.
- Debe visualizarse correctamente en dispositivos con pantallas de 4 a 27 pulgadas, adaptando el diseño mediante CSS responsivo.
- El frontend debe estar desarrollado en React (HTML5, CSS3, JavaScript/TypeScript) y el backend en Spring Boot (Java), facilitando su despliegue en servidores reales.
- No debe depender de librerías o frameworks que requieran instalación adicional en el cliente para su funcionamiento básico.