# Documento de Requerimientos del Sistema

## Proyecto: CarMatch – Plataforma Comparativa de Repuestos y Servicios Automotrices

**Autores:** Mario Francisco Fica Sánchez, Ulises Rosales Alejos  
**Carrera:** Ingeniería Informática  
**Institución:** Duoc UC  
**Documento:** Especificación de Requerimientos de Software (ERS)

## 1. Indice

[**Documento de Requerimientos del Sistema**](#_heading=) **1**

[Proyecto: CarMatch – Plataforma Comparativa de Repuestos y Servicios Automotrices](#_heading=h.bnggje7x2zdf) 1

[1. Introducción](#_heading=) 3

[1.1 Propósito del documento](#_heading=) 3

[1.2 Alcance del sistema](#_heading=) 3

[1.3 Definiciones, acrónimos y abreviaturas](#_heading=) 3

[1.4 Referencias](#_heading=) 3

[2. Descripción General](#_heading=) 3

[2.1 Perspectiva del producto](#_heading=) 3

[2.2 Funciones del sistema](#_heading=) 4

[2.3 Características de los usuarios](#_heading=) 4

[2.4 Restricciones](#_heading=) 4

[2.5 Suposiciones y dependencias](#_heading=) 4

[3. Requerimientos Funcionales](#_heading=) 4

[3.1 Scraping y gestión de datos](#_heading=) 4

[3.2 Base de datos (PostgreSQL)](#_heading=) 4

[3.3 Backend (Django REST)](#_heading=) 4

[3.4 Frontend (Next.js)](#_heading=) 5

[3.5 Administración y control](#_heading=) 5

[4. Requerimientos No Funcionales](#_heading=) 5

[4.1 Escalabilidad](#_heading=) 5

[4.2 Rendimiento](#_heading=) 5

[4.3 Seguridad](#_heading=) 5

[4.4 Disponibilidad](#_heading=) 5

[4.5 Mantenibilidad](#_heading=) 5

[4.6 Usabilidad](#_heading=) 6

[4.7 Portabilidad](#_heading=) 6

[4.8 Monitoreo y métricas](#_heading=) 6

[5. Prioridad y planificación](#_heading=) 6

[6. Aprobación del documento](#_heading=) 6

## 1. Introducción

### 1.1 Propósito del documento

El presente documento tiene como finalidad definir los requerimientos funcionales y no funcionales del sistema **CarMatch**, de acuerdo con las normativas de ingeniería de software y estándares utilizados en la industria (IEEE 830-1998, ISO/IEC 25010). Este documento servirá como guía base para el desarrollo, validación y mantenimiento del sistema.

### 1.2 Alcance del sistema

CarMatch será una plataforma web que permitirá a los usuarios buscar, filtrar y comparar repuestos y servicios automotrices obtenidos a partir de procesos de **web scraping** sobre sitios de terceros. La solución contempla un flujo completo de recolección → almacenamiento en nube (PostgreSQL en GCP) → exposición mediante API REST (Django) → consumo en frontend (Next.js). El sistema estará orientado al mercado chileno de repuestos automotrices.

### 1.3 Definiciones, acrónimos y abreviaturas

* **BD:** Base de datos
* **API REST:** Interfaz de Programación de Aplicaciones basada en arquitectura REST
* **n8n:** Herramienta de automatización utilizada para scraping
* **UI/UX:** Interfaz y experiencia de usuario

### 1.4 Referencias

* IEEE Std 830-1998 – Recommended Practice for Software Requirements Specifications
* ISO/IEC 25010 – Systems and Software Quality Requirements and Evaluation (SQuaRE)
* Normativas internas Duoc UC – Proyecto Portafolio de Título (APT)

## 2. Descripción General

### 2.1 Perspectiva del producto

CarMatch se plantea como un sistema modular compuesto por: - **Módulo de Scraping (n8n):** Recolección de datos en línea. - **Módulo de BD (PostgreSQL/GCP):** Almacenamiento estructurado de la información. - **Módulo Backend (Django):** API REST que expone datos y lógica de negocio. - **Módulo Frontend (Next.js):** Interfaz gráfica web de interacción con el usuario. - **Módulo de Administración:** Panel para gestión de fuentes, usuarios y logs.

### 2.2 Funciones del sistema

* Búsqueda de repuestos y servicios por palabras clave.
* Filtrado avanzado (precio, proveedor, compatibilidad, marca).
* Comparación de productos.
* Visualización de detalle y enlace al proveedor.
* Registro de usuarios y gestión de alertas.
* Panel administrativo para monitoreo y gestión.

### 2.3 Características de los usuarios

* **Usuarios finales:** Conductores y particulares en búsqueda de repuestos.
* **Administradores:** Encargados de la gestión de scraping, BD y monitoreo.

### 2.4 Restricciones

* Dependencia de disponibilidad de sitios web externos (riesgo de bloqueos).
* Cumplimiento de normativa de datos y buenas prácticas de seguridad.

### 2.5 Suposiciones y dependencias

* Se asume conexión estable a internet.
* El usuario final accede desde navegadores modernos.

## 3. Requerimientos Funcionales

### 3.1 Scraping y gestión de datos

* RF-01: El sistema debe permitir configurar fuentes de scraping.
* RF-02: El sistema debe ejecutar scraping automático en intervalos de tiempo definidos.
* RF-03: El sistema debe permitir la ejecución manual de scraping.
* RF-04: El sistema debe manejar bloqueos, timeouts y errores, generando logs.
* RF-05: El sistema debe almacenar fecha y hora de la última actualización de cada producto.

### 3.2 Base de datos (PostgreSQL)

* RF-06: El sistema debe almacenar productos con atributos (nombre, marca, proveedor, precio, stock, compatibilidad).
* RF-07: El sistema debe registrar un historial de precios por producto.
* RF-08: El sistema debe controlar duplicados en la inserción de datos.
* RF-09: El sistema debe permitir consultas complejas con múltiples filtros.

### 3.3 Backend (Django REST)

* RF-10: La API debe exponer endpoints de búsqueda, filtrado y comparación.
* RF-11: La API debe implementar validaciones de seguridad (inyección SQL, XSS, CSRF).
* RF-12: La API debe soportar autenticación de usuarios (tokens o JWT).

### 3.4 Frontend (Next.js)

* RF-13: El sistema debe permitir búsqueda y filtrado visual de productos.
* RF-14: El sistema debe permitir agregar productos a una tabla comparativa.
* RF-15: El sistema debe ser responsivo y usable en móviles.
* RF-16: El sistema debe permitir a los usuarios configurar alertas de precio.

### 3.5 Administración y control

* RF-17: El sistema debe ofrecer un panel de administración con métricas.
* RF-18: El sistema debe permitir activar/desactivar fuentes de scraping.
* RF-19: El sistema debe gestionar usuarios y roles.
* RF-20: El sistema debe exportar reportes en CSV o PDF.

## 4. Requerimientos No Funcionales

### 4.1 Escalabilidad

* RNF-01: El sistema debe permitir la incorporación de nuevas fuentes sin rediseño completo.
* RNF-02: La base de datos debe soportar hasta 10 millones de registros.

### 4.2 Rendimiento

* RNF-03: La API debe responder en menos de 2 segundos en consultas comunes.
* RNF-04: El frontend debe cargar en menos de 5 segundos en móviles.

### 4.3 Seguridad

* RNF-05: El sistema debe implementar cifrado de datos en tránsito (TLS 1.2+).
* RNF-06: El sistema debe cumplir con roles de acceso diferenciados.
* RNF-07: Los datos sensibles deben almacenarse cifrados.

### 4.4 Disponibilidad

* RNF-08: El sistema debe garantizar 99% de disponibilidad mensual.
* RNF-09: La base de datos debe contar con respaldos automáticos diarios.

### 4.5 Mantenibilidad

* RNF-10: El sistema debe contar con pruebas unitarias e integrales.
* RNF-11: El sistema debe estar documentado con manual técnico y de usuario.
* RNF-12: El sistema debe estar versionado con Git.

### 4.6 Usabilidad

* RNF-13: La interfaz debe cumplir con estándares de accesibilidad WCAG AA.
* RNF-14: El sistema debe ser intuitivo, con navegación clara y consistente.

### 4.7 Portabilidad

* RNF-15: El sistema debe poder desplegarse en contenedores Docker.
* RNF-16: La BD debe poder migrarse a AWS RDS o Azure SQL sin cambios críticos.

### 4.8 Monitoreo y métricas

* RNF-17: El sistema debe exponer métricas vía API o dashboards.
* RNF-18: El sistema debe generar alertas en caso de fallos críticos.

## 5. Prioridad y planificación

Cada requerimiento será clasificado en **Alta, Media o Baja** según su impacto en el sistema. La priorización se gestionará mediante metodología ágil (Scrum), iterando en sprints.

## 6. Aprobación del documento

Este documento servirá como referencia oficial para el desarrollo y será actualizado en conjunto con el avance del proyecto. Su aprobación queda a cargo de los autores y del docente guía.