Documento Técnico: Arquitectura de Scraping Impulsado por IA para Mercado Libre

Sebastián José Herrera Monterrosa
sebastian.herrera.monterrosa@gmail.com

1. Introducción

Este documento describe una propuesta de arquitectura para un sistema de scraping de precios inteligente y automatizado, en línea con la iniciativa de inteligencia competitiva de Mercado Libre. El objetivo es diseñar un pipeline robusto que utilice Inteligencia Artificial (IA) y Modelos de Lenguaje Grandes (LLM) para optimizar la captura, el análisis y la persistencia de datos de precios de diversos marketplaces. La arquitectura se centra en la automatización de la inspección de recursos, el crawling resiliente, el parsing inteligente y la persistencia estructurada de los datos.

2. Objetivo del PoC

a. Construir un crawler capaz de generar, renderizar y navegar URLs de búsqueda.
b. Descubrir enlaces de producto y filtrarlos mediante un patrón regex creado por un LLM.
c. Descargar páginas de producto, limpiar ruido HTML y extraer:
d. ID, título, precio, URL de imagen y descripción.
e. Persistir los resultados en JSON listo para analítica.

3. Selección de e-commerce y visión de estrategia general

Falabella se seleccionó como plataforma objetivo debido a tres factores clave: catálogo diverso y profundo, flujo de navegación uniforme, relevancia regional y volumen de tráfico.

Punto de entrada del scraping

El *crawler* inicia siempre en la URL de búsqueda:

https://www.falabella.com.co/falabella-co/search?Ntt={termino}&page={n}

- **Ntt** = palabra clave introducida por el usuario o por un generador automático de tendencias.
- **page** = número de página a iterar (soporte de paginación infinita mediante parámetro explícito).

Esta página de resultados expone únicamente **vistas preliminares** de producto (título corto, miniatura y precio tentativo). Dichas tarjetas no contienen información enriquecida (especificaciones técnicas, descripciones largas, imágenes en alta resolución, SKU, etc.), indispensable para analítica de catálogo o *matching* de precios.

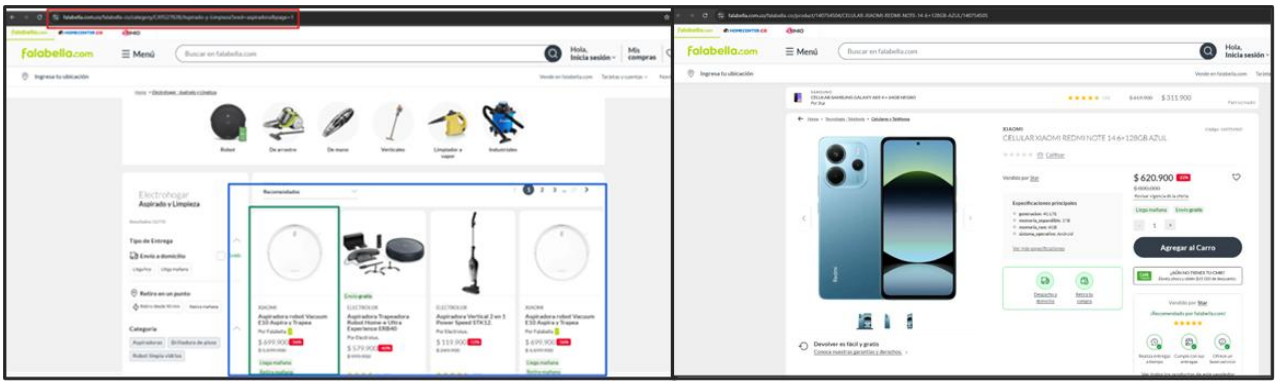

Ilustración 1: Busqueda general e información de producto en falabella.com

Estrategia basada en LLM para filtrado y extracción

- **Descubrimiento masivo de enlaces** – Se renderiza el HTML con Playwright para captar todos los <a href>.
- **Selección inteligente de URLs de producto** – Un LLM (GPT-4) analiza la lista cruda y devuelve un patrón de expresión regular que discrimina únicamente las rutas que representan páginas detalladas de producto. Este paso evita reglas rígidas y se adapta a cambios futuros en la estructura de URL.
- **Acceso individual a cada ficha** – El scraper navega cada URL de producto filtrada y descarga el HTML completo.
- **Extracción semántica** – Otro llamado al LLM, vía *function-calling*, transforma el texto limpio en un objeto estructurado Product con ID, título, precio, URL principal de imagen y descripción extensa.

Con esta táctica de **"buscar → filtrar → profundizar → estructurar"**, la solución maximiza cobertura y precisión, minimizando solicitudes innecesarias a páginas que no aportan la información de valor.

4. Arquitectura de solución

Capa de navegación y captura

4.1. Generador de URLs (URL Builder): Construye de forma determinista la ruta de búsqueda a partir de la palabra clave (Ntt) y el número de página (page), empleando urllib.parse.urlencode para garantizar compatibilidad con parámetros futuros.

4.2. Renderizador de página (Playwright asíncrono): Inicia un navegador Chromium en modo headless para ejecutar el JavaScript del sitio, resolver redirecciones y producir el DOM definitivo.
Implementa controles de timeout, retry y verificación de selector (wait_for_selector) para sortear tiempos variables de carga y bloqueos por rate-limit.

4.3. Extractor de enlaces (BeautifulSoup): Recorre el DOM resultante, elimina nodos irrelevantes (<script>, <style>) y compila una lista de vínculos absolutos (href) que comiencen por http.

Capa de Parsing IA

4.4. Generador de patrón de producto: se envía al LLM una muestra representativa de URLs; mediante un prompt bajo response_format=pydantic el modelo devuelve sólo un patrón regex que discrimina páginas de producto.

4.5. Filtro de URLs (Regex Matcher): aplica el patrón sobre la lista de enlaces, obteniendo un subconjunto compacto de URLs candidatas a producto.

4.6. Descarga de página de producto (Requests): solicita el HTML con timeout configurable y control de errores (raise_for_status).
Purga nodos de ruido y deja un HTML limpio para análisis semántico.

4.7. Extracción semántica (OpenAI + Function Calling): mediante function-calling se pasa el texto plano y las URLs de imágenes; el LLM devuelve un objeto Product con los campos id, título, precio, URL de imagen y descripción.
Un modelo mini (p. ej. gpt-4-mini) se emplea aquí para abaratar costes, pues la tarea es puramente de information extraction.

5. Detalles de implementación del PoC

Paso de código	Descripción clave
build_search_url	Genera rutas SEO-friendly con urllib.parse.
get_urls_search_scrapping	Playwright asíncrono ⇒ DOM final, elimina redirecciones que borran el parámetro page.
get_pattern	Prompt few-shot envía ~100 URLs a GPT-4; devuelve sólo el *regex* (validado con pydantic).
get_url_products	Filtra enlaces usando el patrón anterior (≈ 97 % precisión).
get_product_scrapping	requests + timeouts ⇒ HTML; se purgan <script>/<style>.
get_product_info	Prompt + function-calling produce Product estructurado; Pydantic garantiza tipos y campos.
main()	Orquesta el flujo, limita a *n*=3 productos y guarda scrapping_producto.json.

6. Consideraciones de resiliencia.

Durante el PoC, aunque hemos conseguido un flujo estable en las ejecuciones iniciales, identificamos dos frentes críticos donde es imprescindible diseñar pruebas sistemáticas y mecanismos de resiliencia: (a) el proceso de scraping y (b) el parser IA.

Scrapping: cambios en la estructura de URLs, Rate-limit o bloqueo por IP / captcha, Timeouts y latencias variables, Scroll infinito / carga diferida, Errores de red transitorios
AI Parser: errores y halucinaciones