Central de Reviews - Mejora #1249

Meli Reviews

2026-03-17 22:26 - Jorge Defeo

Estado:	Nuevo	**Fecha de inicio:**	2026-03-17
Prioridad:	Normal	**Fecha fin:**	
Asignado a:		**% Realizado:**	0%
Categoría:		**Tiempo estimado:**	0.00 hora
Versión prevista:		**Tiempo dedicado:**	0.00 hora
Pilar:		**WorkSpace:**	
Valor de la Tarea:		**Contacto :**	

Descripción

Objetivo:
A partir de un producto, buscar en el catalogo de mercadolibre y extraer las reviews.
Almacenarlas en masterdata v2, para luego mostrarlas en cada producto.

Es un enfoque mucho más universal. Partir desde el EAN (código de barras) es la forma correcta de hacerlo si vas a actuar como un agente externo, ya que te independizas de un vendedor en particular y puedes auditar la performance general del producto a nivel catálogo.

Para lograr esto, no puedes pegarle directamente al endpoint de opiniones con el EAN. Necesitas hacer un "puente" de dos pasos consultando primero el buscador de Mercado Libre.

Aquí tienes la lógica técnica detallada y un script de Python listo para que lo corras o se lo pases a Antigravity para que te lo extienda.

1. La Lógica del Flujo (EAN -> Opiniones)
Resolver el GTIN (EAN): Le pegas a la API de búsqueda (usaremos el site_id MLA para Argentina) pasando el EAN en el parámetro gtin.

Extraer los IDs: De los resultados, tomas la primera publicación que aparezca y le extraes el id (el ID de publicación) y, lo más importante, el catalog_product_id.

Paginación de Reviews: Le pegas al endpoint de /reviews/item/ enviando ambos IDs. Como Mercado Libre devuelve las opiniones paginadas (por defecto trae solo 5), tienes que armar un loop iterando sobre el offset hasta que se agoten para poder traerlas todas.

2. Script Base en Python
Este código hace el flujo completo de forma autónoma. Solo necesita la librería requests.

Python
import requests

def obtener_todas_las_reviews(ean, token):
headers = {"Authorization": f"Bearer {token}"}

1. Paso 1: Buscar la publicación en MLA usando el EAN (GTIN)
   url_search = f"https://api.mercadolibre.com/sites/MLA/search?gtin={ean}"
   res_search = requests.get(url_search, headers=headers).json()

```
if not res_search.get("results"):
        print(f"No se encontraron publicaciones activas para el EAN {ean}")
        return []
```

1. Paso 2: Extraer los IDs del primer resultado
   item = res_search["results"][0]
   item_id = item["id"]
   catalog_product_id = item.get("catalog_product_id")

1. Paso 3: Armar la URL del endpoint de opiniones
   url_reviews = f"https://api.mercadolibre.com/reviews/item/{item_id}"
   if catalog_product_id: # Fundamental para traer el acumulado de TODO el catálogo

```
    url_reviews += f"?catalog_product_id={catalog_product_id}"

reviews_totales = []
    offset = 0
    limit = 50 # Tamaño del batch por petición
```

1. Paso 4: Loop de paginación para agotar todas las opiniones
   while True:
   params = {"limit": limit, "offset": offset}
   res_reviews = requests.get(url_reviews, headers=headers, params=params).json()

   batch = res_reviews.get("reviews", [])
   if not batch: # Si viene vacío, ya llegamos al final
   break

   reviews_totales.extend(batch)
   offset += limit

```
return reviews_totales
```

1. --- Ejecución ---
2. EAN_DE_PRUEBA = "7791234567890"
3. TOKEN_MELI = "APP_USR-..."
4. data_reviews = obtener_todas_las_reviews(EAN_DE_PRUEBA, TOKEN_MELI)
5. print(f"Se extrajeron {len(data_reviews)} opiniones exitosamente.")
   3. Consideraciones para Almacenar la Data
   Si el objetivo es ir guardando este histórico para cruzar datos o auditar, te recomiendo modelar la inserción directa en tu base de datos (por ejemplo, en Supabase).

De los diccionarios que te devuelve la lista data_reviews, los campos claves que deberías mapear a las columnas de tu base son:

id (BigInt, ideal como Primary Key).

rate (Integer 1-5).

title (Texto corto).

content (Texto largo, la opinión completa).

likes / dislikes (Integers, fundamentales para saber si la review es relevante para los usuarios).