DUPLICA EL ARCHIVO Y PRESENTA SOBRE EL NUEVO DOCUMENTO

Desafío Técnico — RataLibre

Objetivo general

Desarrollar una aplicación que se conecte a **Mercado Libre**, consuma la **API oficial**, obtenga la información completa de **una publicación existente del seller**, la persista en una **base de datos tipo Supabase**, y utilice un **LLM (OpenAI)** para generar **recomendaciones accionables para el e-commerce**.

El foco del desafío no es solo el resultado final, sino **el proceso de desarrollo asistido por IA**, que debe quedar completamente auditado a través de **commits incrementales en GitHub**.

Flujo funcional esperado (end-to-end)

1. Conexión a Mercado Libre

- Implementar OAuth para conectar una cuenta de Mercado Libre (usuario test o cuenta propia).
- Obtener y guardar el `access_token` **solo del lado servidor**.
- No exponer tokens en frontend ni en commits.
- https://developers.mercadolibre.com.ar/mcp-server

---

2. Lectura de datos desde la API

La app debe traer, **como mínimo**, los datos de **una publicación existente** del seller:

- Datos del ítem:

    - item_id
    - título
    - precio
    - estado
    - stock disponible
    - vendidos
    - categoría
- Descripción del producto
- (Opcional) atributos principales

La lectura debe hacerse usando la **API oficial de Mercado Libre**.

---

3. Persistencia en base de datos

- Guardar toda la información obtenida en una base de datos tipo **Supabase** (Postgres).
- Debe existir al menos:
    - tabla de publicaciones
    - tabla de descripciones / metadata
- El modelo debe permitir volver a correr el análisis sin volver a pegarle a la API.

---

4. Procesamiento + LLM

Una vez persistidos los datos:

1. Tomar:
    - datos estructurados del ítem
    - descripción completa
    - métricas simples calculadas (ej: precio, stock, estado)
2. Armar un payload claro y acotado.
3. Enviar esa información a un **LLM ej OpenAI**, usando la API oficial (keys por env vars).
4. El LLM debe devolver **recomendaciones para el e-commerce**, mediante un prompt generico desarrollado con ChatGPT, y todoa la logica que entregue:
    - mejoras de título
    - problemas en la descripción
    - oportunidades de conversión
    - riesgos comerciales

---

6. UI mínima

- Vista donde se muestre:
    - la publicación cargada (en tabla o card)
    - el output del LLM
- Botón: **“Analizar publicación”**

No se evalúa diseño visual, sí claridad.

Stack (flexible)

- Frontend / Fullstack: Next.js, React, o equivalente
- Backend: Node.js
- DB: Supabase o Postgres equivalente
- LLM: OpenAI API
- IDE: **Cursor (obligatorio)**

Restricción clave: desarrollo 100% auditable

Repo GitHub (obligatorio)

- El candidato debe crear un repo desde el inicio.
- El repo debe contener **todo el historial del desarrollo**.

Regla central: commit por prompt

👉 Cada vez que el candidato hace una pregunta relevante al AI en Cursor y eso genera cambios, debe:

1. Aplicar los cambios
2. Hacer un commit inmediato
3. Dejar registro del prompt

No se permiten:

- Mega commits
- “feat: proyecto terminado”
- Commits sin relación clara con una acción concreta

Qué se evalúa

Técnico

- Correcto uso de la API de Mercado Libre
- Persistencia real en DB
- Uso correcto de OpenAI API
- Separación de responsabilidades
- Manejo de env vars

Proceso (clave)

- Uso real de Cursor + AI
- Calidad de prompts
- Iteración incremental
- Claridad en commits
- Capacidad de corregir errores

Producto

- Recomendaciones útiles

Demo obligatoria

En la demo final, el candidato debe mostrar:

1. El repo y el historial de commits
2. Una publicación real cargada