RP-01 | Protocolo de investigación | VERSION 0.2 | AUGUST 2026

Extracción multi-etapa con LLM y procedencia restringida para catálogos heterogéneos: diseño y protocolo de benchmark

Pablo Cardozo
Investigador independiente
Buenos Aires, Argentina

Resumen

Los catálogos de productos en PDF y planillas combinan layout visual, descripciones abreviadas, convenciones de packaging, precios, descuentos y límites de fila inconsistentes. Un único prompt puede devolver JSON válido y aun así omitir filas, fusionar productos o inventar procedencia. Este protocolo estudia un pipeline implementado de tres etapas que separa metadata documental, reconstrucción de filas fuente y normalización de productos. Cada etapa está restringida por schemas. La etapa final procesa lotes de diez filas y debe repetir una identidad de lote determinista, el conjunto exacto de identificadores y la cobertura de páginas antes de persistir. El post-procesamiento fuerza a revisión humana los productos ambiguos, de baja confianza o no normalizables. El 4 de agosto de 2026 se fijó la revisión registrada y la suite local completa aprobó 569 tests en 20 archivos en 1,06 segundos. También aprobaron cinco tests de performance, pero simulan inserción con un array en memoria y no aportan evidencia sobre throughput de Convex, del modelo ni end-to-end. No existe en los artefactos inspeccionados un corpus etiquetado, baseline de una pasada, predicciones crudas ni resultado de exactitud. El aporte es doble: auditoría reproducible de los controles de procedencia y diseño falsable para comparar el sistema con un baseline fijado. Las métricas separan detección de filas, normalización, filas inventadas, atribución, carga de revisión, latencia y costo. No se afirma que la descomposición mejore exactitud hasta ejecutar el experimento.

Palabras clave: extracción documental; generación estructurada; procedencia; line items; revisión humana; protocolo de evaluación

ESTADO DE LA EVIDENCIA	Arquitectura y 569 tests de software reproducidos; benchmark de extracción aún sin ejecutar
GRADO DE EVIDENCIA	Grado S/P — evidencia de software reproducida y protocolo experimental sin ejecutar
REVISIÓN FUENTE	a5d3aadbfa04
REPRODUCIDO / AUDITADO	2026-08-04

Datos rastreables

SUITE LOCAL	569/569 tests · 20 archivos
DURACIÓN	1,06 s con Node 24.14.0
PIPELINE	3 etapas · lotes de 10 filas
EXACTITUD	No medida en corpus etiquetado

Contenido

> **LÍMITE EDITORIAL**
> Las afirmaciones quedan acotadas a la revisión citada y al grado de evidencia. Las mediciones ausentes se declaran como ausentes; no se infieren de la calidad de implementación.

1. Formulación del problema y contribución

Un catálogo de proveedor no es solo un conjunto de strings. Sus filas codifican producto, marca, presentación, formato de venta, unidad, precio y a veces descuento o condición mayorista. El significado se reparte entre encabezados, celdas combinadas, grupos visuales, notas y abreviaturas. Las planillas exponen celdas pero pueden tener filas decorativas y registros multilínea; los PDF preservan apariencia y pierden semántica tabular. La extracción debe recuperar campos y un vínculo defendible con la fila fuente.

La implementación Cuqui separa metadata documental, reconstrucción de filas y normalización [1–3]. La idea es plausible: las etapas tempranas preservan layout e identidad; las posteriores trabajan con contexto acotado y schema estricto. Plausibilidad no es evidencia comparativa. La pregunta es si la estructura reduce omisiones, alucinaciones y revisión frente a un prompt de una pasada manteniendo constantes modelo, documentos y scoring.

- Especificar la tarea en niveles de fila, campo y procedencia.
- Auditar etapas, validaciones, retries y gates de revisión.
- Reproducir los tests deterministas en una revisión fijada.
- Diseñar una comparación controlada sin afirmar un resultado no ejecutado.
- Definir artefactos suficientes para scoring y análisis independiente.

> **LÍMITE DE LA AFIRMACIÓN**
> 569 tests aprobados establecen las propiedades codificadas por esos tests. No establecen exactitud de extracción en catálogos reales.

2. Trabajo relacionado y posicionamiento

Document AI aborda documentos visuales mediante modelos multimodales con OCR y sistemas image-to-sequence sin OCR. LayoutLMv3 preentrena texto e imagen para tareas documentales [8]. Donut elimina el OCR externo y mapea imágenes a salida estructurada [7]. Nougat convierte páginas científicas a markup preservando semántica perdida en PDF [10]. Esos son modelos aprendidos; el sistema presente orquesta un modelo multimodal general mediante etapas y schemas de negocio.

DocILE es la referencia de benchmark más cercana porque separa extracción de información y reconocimiento de line items, con documentos anotados, baselines y layouts diversos [9]. Los catálogos no son facturas, pero la lección transfiere: hay que identificar line items antes de puntuar campos y reservar layouts no vistos, no mezclar filas del mismo template entre splits.

Enfoque	Entrada	Salida	Rol
LayoutLMv3-like	OCR + layout + imagen	Labels específicos	Contexto, no baseline implementado
Donut-like	Imagen documental	Secuencia estructurada	Contexto, no baseline implementado
LLM una pasada	Catálogo completo	JSON de productos	Baseline principal
LLM tres etapas	Archivo, filas y lotes	JSON con procedencia	Sistema estudiado

Tabla 1. Posición del benchmark.

3. Definición de tarea y unidad de análisis

Sea D un documento con filas fuente ordenadas R. Una anotación gold mapea cada fila válida a cero o un producto normalizado con nombre, marca, categoría, packaging, tipo de precio, monto, unidad y decisión de revisión. Encabezados, totales y términos son no-producto. Una predicción debe identificar primero la fila y después asignar campos. Así no se premian productos plausibles sin origen rastreable.

Nivel	Pregunta	Falla
Fila	¿Se recuperaron todos y solo los productos?	Un ítem multilínea se divide
Campo	¿Se normalizaron valores?	12 × 500 g tratado como 500 g total
Procedencia	¿El producto apunta a la fila real?	Producto correcto con rowId ajeno
Decisión	¿La ambigüedad fue a revisión?	Precio dudoso auto-publicado

Tabla 2. Tres niveles de corrección.

La unidad es una fila gold anidada en documento. Se reportan macro-promedios por documento y micro-promedios por fila para que una planilla grande no domine. Los campos se puntúan después de emparejar por procedencia. Predicciones sin match son filas inventadas; gold sin match son omisiones.

4. Pipeline implementado de tres etapas

El pipeline valida y sube el archivo, espera estado activo y ejecuta dos etapas documentales antes de normalizar lotes [2]. Stage 1 y 2 usan gemini-3.1-pro; Stage 3 usa gemini-3.1-flash-lite-preview en la revisión registrada. Cada generación solicita JSON con JSON Schema, timeout de 60 s y hasta tres intentos con backoff lineal [2]. Los modelos son sensibles a versión y deben fijarse de nuevo al experimentar.

Etapa	Entrada	Salida	Gate determinista
1 — metadata	Archivo completo	Páginas, layout y regiones	Parseo de schema
2 — filas	Archivo + Stage 1	Filas con rowId	Schema, total y IDs únicos
3 — productos	Diez filas + metadata	Productos + contexto de lote	Identidad, pertenencia y páginas
Post-proceso	Extracción	Producto o error	Precio, confianza, unidad y revisión

Tabla 3. Responsabilidades y gates.

Stage 2 es la bisagra: el total aplanado debe igualar totalRowCount y cada rowId ser único. Stage 3 agrupa de a diez. Un batchId determinista incluye índice y primer/último rowId; el modelo debe devolver el mismo ID, índice, total, rowIds ordenados y páginas. Un item con sourceRowId fuera del lote rechaza el resultado [2,3].

Listado 1. Flujo simplificado.

```
metadata = stage1(file)
rows = stage2(file, metadata)
assert unique(rows.rowId)
assert flatten(rows).length == rows.totalRowCount
for batch in chunk(rows, 10):
    result = stage3(metadata, batch)
    assert result.context == expected_context(batch)
    assert every(result.item.sourceRowId in batch.rowIds)
    normalize_or_route_to_review(result.items)
```

5. Procedencia y gates de revisión humana

JSON válido es necesario pero insuficiente. La implementación agrega checks relacionales: rowIds únicos, identidad exacta del lote, pertenencia ordenada, páginas y sourceRowId [2,3]. Previenen respuestas de otro lote, filas estructuralmente válidas pero desalineadas y productos sin origen en el contexto.

Trigger	Regla	Razón
Baja confianza	confidence < 0,5	No auto-publicar incertidumbre
Sin packaging	Objeto ausente	Presentación y precio normalizado inseguros
Normalización fallida	No se calcula precio unitario	Evitar precios incomparables
Nombre genérico	Vacío, unknown o <3 caracteres	Evitar entradas inútiles
Monto inválido	Precio ≤ 0	Rechazar
Fila inválida	sourceRowId fuera del lote	Rechazar respuesta

Tabla 4. Triggers deterministas de revisión.

El umbral 0,5 es política, no probabilidad calibrada. La autoconfianza del modelo puede derivar. El benchmark debe medir exactitud selectiva: entre ítems auto-aprobados, cuántos son correctos y qué proporción se difiere. Así el gate se vuelve regla evaluada.

```
exactitud_selectiva(τ) = correctos_autoaprobados(τ) / autoaprobados(τ)
```

Debe leerse junto con cobertura(τ). Alta exactitud con cobertura casi cero no es útil.

6. Reanudación, duplicados y estado operativo

La action valida magic bytes y tamaño, calcula SHA-256 y rechaza duplicados recientes por proveedor [2]. Guarda identidad del archivo, JSON intermedio, progreso, lotes, errores y resultados. Stage 3 puede reanudar desde el primer lote cuyas filas no existen. Los temporales se borran en finally. Estos controles añaden variables experimentales: retries, resumes y duplicados deben aparecer en metadata.

Evento	Comportamiento	Campo requerido
Procesamiento	Poll cada 2 s, hasta 120	poll_count, activation_ms
Timeout	60 s por intento	stage, attempt, timeout
Retry	Hasta 3	error e intentos
Falla de lote	Índice almacenado	IDs y razón
Resume	Salta filas existentes	Punto y run anterior
Duplicado	Proveedor + SHA-256	Decisión

Tabla 5. Eventos operativos a registrar.

7. Evidencia de software reproducida

Se clonó el repositorio y fijó a5d3aadbfa049f4f6d77a2f144f557fa9bf7cffb. Con Node 24.14.0 y npm 11.9.0, npm ci completó y npm test salió cero. Vitest reportó 20 archivos y 569 tests aprobados en 1,06 s, con 693 ms de tests. El conteo reemplaza para esta revisión la cifra 409 del README [1,5].

Listado 2. Comando de reproducción.

```
git checkout --detach a5d3aadbfa049f4f6d77a2f144f557fa9bf7cffb
npm ci
npm test
```

Observación	Conclusión sostenida	No sostenida
569 tests	Validadores y comportamientos codificados pasan	Existen 569 ejemplos de research
20 archivos	Cobertura mayor al README	Se contactaron integraciones productivas
1,06 s	Suite determinista rápida	La ingesta tarda 1,06 s
Revisión fijada	Resultado atado a código	El modelo futuro no cambia

Tabla 6. Qué establece la suite.

El archivo de performance también aprobó cinco tests. Genera 10.000 mocks y empuja IDs a un array, imitando spread de objetos [6]. No llama Convex, red, Gemini ni persiste. Por eso no se usa la interpretación de millones de productos por segundo del documento de decisiones [4]. Solo se sostiene que la simulación y assertions completaron.

> **LÍMITE BRUTAL DE EVIDENCIA**
> Un loop en memoria no es benchmark de base de datos y una suite aprobada no es estudio de exactitud.

8. Diseño del benchmark controlado

El piloto propuesto usa al menos 60 catálogos autorizados: 20 PDF, 20 XLS y 20 XLSX. Se estratifica por layout, filas, multilínea, ambigüedad de packs, unidades mixtas, descuentos y escaneos. Documentos del mismo template quedan en un mismo split. Desarrollo se usa para iterar prompts; test se abre una sola vez.

Condición	Descripción	Controles
B0 — parser	Parser de celdas para XLS/XLSX	Mismas filas y scorer
B1 — una pasada	Un prompt devuelve JSON completo	Mismo modelo, archivo y ontología
S1 — tres etapas	Metadata → filas → productos	Misma fuente y schema final
S1 sin review	Sin regla de revisión forzada	Mide aporte del gate

Tabla 7. Condiciones experimentales.

Cada par documento-condición corre al menos tres veces. La comparación es pareada por documento. Se archivan modelos, prompts, schemas, temperatura, retries y precios. Una falla después de retries cuenta como documento fallido; excluirla sesgaría exactitud y latencia.

1. Definir guía y etiquetar antes de ver outputs.
2. Etiquetar por duplicado al menos 20% y adjudicar.
3. Congelar splits, prompts, schemas, modelos y scorer.
4. Ejecutar condiciones idénticas y registrar retries.
5. Publicar gold redactado, predicciones, matches, agregados y fallas.

9. Métricas y análisis estadístico

```
precisión_fila = predicciones_emparejadas / predicciones
```

Penaliza filas inventadas o duplicadas.

```
recall_fila = filas_gold_emparejadas / filas_gold
```

Penaliza productos omitidos; reportar F1 sin ocultar precision/recall.

```
exactitud_campo_f = correctos_f / filas_emparejadas_con_f
```

Se calcula tras matching de procedencia con reglas públicas de normalización.

```
error_procedencia = sourceRowId_incorrecto / predicciones
```

Un valor textual correcto falla si está unido a otra fila.

Familia	Medidas
Extracción	Precision/recall/F1 de filas y exactitud de campos
Procedencia	IDs inválidos, matches erróneos, páginas y duplicados
Revisión	Tasa, exactitud selectiva, errores aprobados y correctos diferidos
Confiabilidad	Éxito por documento, retries, timeouts y resumes

Familia	Medidas
Operación	Latencia, tokens, costo y procesamiento del archivo

Tabla 8. Familias de outcomes.

Se reportan medias macro por documento con intervalos bootstrap 95%. Para outcomes binarios pareados puede usarse McNemar; para medidas continuas, diferencias bootstrap pareadas. Los campos forman una familia y deben interpretarse con tamaños de efecto, no seleccionando p-values. Los ejemplos se muestrean con taxonomía predeclarada.

10. Taxonomía de error, validez y ética

Código	Error	Ejemplo
E-R1	Fila omitida	Desaparece un producto
E-R2	Fila inventada	Un header se vuelve producto
E-R3	Split/merge	Multilínea dividido o dos filas unidas
E-F1	Transcripción	Marca o precio incorrecto
E-F2	Normalización	Pack o unidad mal calculada
E-P1	Procedencia	Valor correcto unido a otra fila
E-D1	Decisión	Error aprobado o correcto diferido

Tabla 9. Taxonomía propuesta.

- Constructo: JSON válido y cantidad de tests no equivalen a exactitud.
- Interna: las etapas usan modelos distintos; una mejora no se atribuye solo a descomposición.
- Externa: catálogos argentinos de alimentos no representan otros sectores.
- Temporal: modelos preview y precios pueden cambiar.
- Anotación: límites de fila y packaging requieren adjudicación documentada.

Los catálogos pueden contener identidades, precios negociados, contactos o términos comerciales. Publicar requiere permiso o redacción irreversible. Hashes, archivos, uploads, logs y ejemplos necesitan política de retención y acceso. Los anotadores deben ver solo lo necesario y el corpus no debe permitir reconstruir combinaciones confidenciales.

11. Conclusión

El sistema contiene una arquitectura razonada: etapas explícitas, generación con schema, identidades de fila y lote, checks de procedencia, reglas de review, retries y estado reanudable. La suite completa es reproducible con 569 tests aprobados. Son resultados legítimos de ingeniería.

La afirmación de research sigue abierta. No hay corpus etiquetado ni comparación de una pasada; no se demostró mejor exactitud. Este protocolo define el experimento faltante y evita sustituir calidad de modelo por volumen de tests o throughput en memoria. Ejecutarlo y liberar resultados es el próximo paso.

12. Referencias y reproducibilidad

[1] Cardozo, P. cuqui, revision a5d3aadbfa049f4f6d77a2f144f557fa9bf7cffb, 2026. Repositorio público Cuqui

[2] Cardozo, P. convex/ingest.ts, revision a5d3aadbfa049, 2026. Implementación de ingesta en tres etapas

[3] Cardozo, P. convex/lib/schemas.ts, revision a5d3aadbfa049, 2026. Schemas y prompts de extracción

[4] Cardozo, P. Implementation Decisions — Cuqui v1.0, 30 March 2026. Decisiones y limitaciones conocidas

[5] Cardozo, P. Cuqui README and Tech Inventory, revision a5d3aadbfa049, 2026. README e inventario de tests

[6] Cardozo, P. tests/performance/batch-throughput.test.ts, revision a5d3aadbfa049, 2026. Test de throughput en memoria

[7] Kim, G. et al. OCR-free Document Understanding Transformer. arXiv:2111.15664, 2021. OCR-free Document Understanding Transformer (Donut)

[8] Huang, Y. et al. LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. arXiv:2204.08387, 2022. LayoutLMv3: Pre-training for Document AI

[9] Šimsa, Š. et al. DocILE Benchmark for Document Information Localization and Extraction. arXiv:2302.05658, 2023. DocILE Benchmark for Document Information Localization and Extraction

[10] Blecher, L. et al. Nougat: Neural Optical Understanding for Academic Documents. arXiv:2308.13418, 2023. Nougat: Neural Optical Understanding for Academic Documents

Revisión fuente: a5d3aadbfa049f4f6d77a2f144f557fa9bf7cffb

Apéndices

A. Registro mínimo de anotación

Listado A1. Schema conceptual gold.

```
{
  document_id, page_number, source_row_id, raw_text,
  is_product_row, canonical_name, brand, category, subcategory,
  packaging: { type, units_per_pack, net_quantity, net_unit },
  price: { amount, currency, type },
  ambiguity_codes, annotator_id, adjudication_status
}
```

B. Checklist de release

- Corpus autorizado y versionado con splits por template.
- Guía, estadística de doble etiqueta y adjudicación.
- Prompts, schemas, modelos, dependencias, precios y retries fijados.
- Predicciones crudas de todas las corridas, incluidas fallas.
- Scorer abierto para matching, normalización y procedencia.
- Agregados por documento, intervalos, costo, latencia y errores.

Cita sugerida

Cardozo, Pablo. "Extracción multi-etapa con LLM y procedencia restringida para catálogos heterogéneos: diseño y protocolo de benchmark." Protocolo de investigación, version 0.2, 2026. https://pablo.cardozo.com.ar/es/research/multi-stage-document-extraction.

Documento de trabajo independiente. Sin peer review.