ISSN: 1692-7257 - Volumen 2 – Número 48 - 2026

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v2i48.4320

Sistema multiagente para validación de pedidos de materia prima basado en un modelo visión-lenguaje

Multi-agent system for raw material order validation based on a vision-language model

MSc. Anny Astrid Espitia Cubillos[1], PhD. Robinson Jiménez-Moreno[2], Ing. Mateo Pulido Aponte[3]

[1]***Universidad Militar Nueva Granada,*** *Profesora asociada, Facultad de ingeniería, Programa de Ingeniería Industrial, Bogotá, Colombia*
[2]***Universidad Militar Nueva Granada,*** *Profesor asociado, Facultad de ingeniería, Programa de Ingeniería Mecatrónica, Bogotá, Colombia.*
[3]***Universidad Militar Nueva Granada,*** *Asistente de investigación, Facultad de ingeniería, Programa de Ingeniería Industrial, Bogotá, Colombia.*

Correspondencia: *anny.espitia@unimilitar.edu.co*

Recibido: *24 marzo 2026.* ***Aceptado:*** *12 junio 2026.* ***Publicado:*** *06 julio 2026.*

Cómo citar: *D. A. Landinez Quintero, F. E. Moreno García, and W. Palacios Alvarado, "Sistema multiagente para validación de pedidos de materia prima basado en un modelo visión-lenguaje", RCTA, vol. 2, n.º. 48, pp. 57–68, jul. 2026. Recuperado de https://ojs.unipamplona.edu.co/index.php/rcta/article/view/4320*

Resumen: En este documento se presentan los resultados de la integración del modelo Deepseek OCR mediante un modelo de visión a lenguaje y un sistema multi agente con el objetivo de desarrollar un sistema automático de validación de pedidos, el cual opera al interpretar y comparar documentos referentes a un pedido de materia prima como son la factura, la orden de compra y la remisión, integrado a una interfaz web para la carga y procesamiento de los documentos y visualización de resultados. Permitiendo obtener métricas de similitud y completitud del sistema conjunto para facilitar el análisis de múltiples documentos y encontrar inconsistencias en los mismos, en el momento de la recepción y gestión de un pedido. Para la validación se realizaron 35 pruebas usando información real de una empresa dedicada a la fabricación de productos de aseo obteniendo una ejecución exitosa del flujo de interpretación y comparación de información en el 82,353% de los casos.

Palabras clave: modelo Deepseek OCR, sistema multiagente, verificación de documentos, pedidos, materia prima, modelo de visión a lenguaje.

Abstract: This document presents the results of integrating the Deepseek OCR model with a vision-to-language model and a multi-agent system to develop an automated order validation system. This system operates by interpreting and comparing documents related to a raw material order, such as invoices, purchase orders, and delivery notes. It is integrated with a web interface for uploading and processing documents and viewing results. This allows for obtaining accuracy metrics for the overall system, facilitating the analysis of multiple documents, and identifying inconsistencies during order receipt and management. For validation, 35 tests were conducted using real data from a company that manufactures cleaning products. The information interpretation and comparison workflow were

successfully executed in 82.353% of the cases.

Keywords: Deepseek OCR model, multi-agent system, document verification, orders, raw materials, vision-language model.

1. INTRODUCCIÓN

La adquisición de materia prima e insumos de producción requiere la verificación de los pedidos en el momento de su recepción, dicha actividad es importante para validar la coherencia entre documentos clave como las órdenes de compra, pedidos, remisiones y facturas de proveedores. Es común que esta actividad se haga de forma manual, lo que da lugar a errores humanos que no permiten detectar inconsistencia y dan lugar a reprocesos, en especial cuando se validan simultáneamente documentos electrónicos con físicos en formatos diversos. Por ello el presente artículo expone el desarrollo de una solución multiagente con inteligencia artificial para llevar a cabo la verificación de documentos de forma digital mejorando precisión y eficiencia en su ejecución.
Un agente de inteligencia artificial (IA) es capaz de percibir, planificar, descomponer, actuar y reflexionar [1].

Un Sistema Multiagente (MAS) puede entenderse como una red de agentes inteligentes autónomos que interactúan para lograr un objetivo común, resultan útiles para afrontar problemas complejos al dividirlos en subproblemas más sencillos, permitiendo el desarrollo de sistemas resilientes, adaptativos y escalables que permiten la integración de técnicas de inteligencia artificial para mejorar sus capacidades computacionales y de toma de decisiones [2], pese a su potencial afrontan restricciones de hardware. El uso de sistemas multiagentes constituye una opción de automatización eficiente para problemas complicados en áreas logísticas, gracias a la toma descentralizada de decisiones y cooperación entre agentes, que permite respuestas que se adaptan a los cambios operativos [3]. Para revisar los documentos de planeación de plantillas de redes, en [1] usa una arquitectura multiagente basada en modelos de lenguaje de código abierto, lo que mejora su procesamiento y análisis.

Los sistemas multiagentes que usan aprendizaje por refuerzo (MARL) han exhibido contribuciones importantes para la gestión óptima de inventarios [4], aún en condiciones de complejidad como las redes con múltiples niveles [5]. Usando MARL con redes neuronales de grafos para el modelamiento de las relaciones entre los eslabones de las cadenas de suministro, se logran decisiones más robustas que propenden por el trabajo colaborativo [6]. MARL también se ha empleado para mejorar el manejo de los inventarios en escenarios con incertidumbre aleatoria obteniendo respuestas eficientes similares o mejores que las alcanzadas mediante el uso de heurísticas tradicionales [7] y validadas con eficacia mediante simulaciones con condiciones comerciales reales. Sin embargo, estos modelos tienen dificultades asociadas con la estabilidad y la escalabilidad en particular en entornos probabilísticos.

Para el análisis de texto en documentos, el Reconocimiento Óptico de Caracteres (OCR) ha demostrado un rendimiento prometedor, pero con desventajas asociadas a los altos costos computacionales, la falta de flexibilidad en idiomas o tipos de documentos y la propagación de errores del OCR al proceso posterior [8]. Por eso en [9] y en [10] acuden a usar aprendizaje profundo con elementos de tipo generativos logrando ventajas de precisión y eficiencia computacional frente a los OCR tradicionales. También se han integrado tecnologías OCR y de detección de objetos (DO) para reconocimiento de información textual y no textual en etiquetas impresas para aplicaciones industriales [11], su ventaja radica en la extracción de diversos datos. Sin embargo, los errores se propagan a través de cada fase. Los modelos de visión OCR solo se concentran en la transcripción de documentos y no su entendimiento.

Actualmente modelos más avanzados como los modelos de visión a lenguaje (VLM), integran la información de imágenes y textos en un mismo proceso revolucionando la interpretación automática de archivos. Estos modelos mejoran la comprensión de imágenes con texto en distintos formatos como formularios, tablas y documentos escaneados [12].

Existen diversos sistemas multiagente con aplicación en el sector manufacturero. En [13] se propone gestionar el aumento de la producción mediante un sistema multiagente (MAS) que evalúa estrategias en contextos de producción

multivariante y se ilustra su aplicación en el sector de fabricación de muebles, señalando que los MAS coordinan tareas industriales complejas. En [14] se incorporan modelos de lenguaje largo (LLM) a un sistema multiagente para gestionar los recursos de fabricación, sistema que puede implementarse en diferentes plantas de producción inteligentes, reemplazando heurísticas por procesos de negociación en lenguaje natural. Por esta misma vía y dado que la gestión de la cadena de suministro requiere de consensos rutinarios, en [15] se usaron agentes LLM para su automatización, donde los autores validaron su eficacia mediante un estudio de caso de gestión de inventarios donde se redujo el efecto látigo (ampliación de las variaciones de la demanda a través de la cadena de abastecimiento) de una mejor manera que con políticas de reposición de existencias y los enfoques de demanda centralizados.

Pese a los avances referidos, aún existen brechas en la literatura relacionadas con soluciones que integren sistemas multiagentes con VLM, que funcionen adecuadamente en escenarios reales, para interpretar documentos heterogéneos relacionados con la recepción de pedidos de materia prima y validar la consistencia lógica de la información contenida en los mismos automáticamente y en tiempo real. Es en este marco donde el trabajo expuesto en este artículo propone una solución mediante el diseño y validación con datos reales, de una microempresa industrial, por medio de un sistema multiagente para la revisión de pedidos de materia prima basado en un VLM que integra un OCR y agentes inteligentes para, no solo extraer datos de múltiples fuentes, sino para clasificar, interpretar y comparar información de los documentos. La validación documental se realiza empleando métricas de similitud textual y numérica, incluyendo una interfaz web de usuario para visualización de reportes y gestión. El sistema reduce la necesidad de intervención humana y elimina la propagación de errores en la recepción de pedidos de materias primas.

2. METODOLOGÍA

En el diseño y desarrollo del sistema multiagente para validación de pedidos de materia prima se optó por una orientación de investigación aplicada, tipo experimental con enfoque en el diseño, aplicación y validación, para lo cual se establecieron cinco fases.

En la primera fase se diseña la arquitectura del sistema multiagente que integra un sistema OCR, donde se emplea un agente que interpreta y compara la información mediante una interfaz gráfica de usuario.

En la segunda fase se configura y ejecuta el sistema OCR en un entorno local soportado en CUDA, con compatibilidad especifica de la tarjeta gráfica (RTX 5070), que requiere la disposición previa de las librerías Transformers y Pytorch con soporte para la arquitectura Blackwell – sm_120, lo que permite instalar correctamente el modelo de DeepSeek VLM que para segmentar los documentos se usa el modelo SAM de Meta y un decodificador visual (DeepSeek encoder) para entender los tokens visuales, lo que optimiza el procesamiento de los documentos escaneados. Su funcionamiento se valida en dos etapas, la primera con pipelines predefinidos y la segunda con un pipeline optimizado para inicializar los modelos de segmentación y visión en condiciones controladas, lo que permite reducir el uso de memoria y los requisitos computacionales.

En la tercera fase se diseña la arquitectura del sistema multiagente usando el Agent Developer Kit (ADK) de Google [16], con una estructura de agentes tanto secuenciales como en bucle, que integra el modelo de lenguaje grande de OpenAI (GPR-4.1-mini) para clasificar documentos, interpretar los datos y validar la consistencia de información entre documentos [17]. El control del proceso se hace usando callbacks, que permiten monitorear cada paso, conservar resultados intermedios, actualizar instrucciones y finalizar el proceso. Se diseña una estructura para contar con una salida de interpretación consistente entre los documentos: factura, orden de compra y remisión.

En la cuarta fase se diseña la interfaz web de interacción del sistema multiagente con los usuarios en Streamlit, dado que permite la implementación de interfaces sencillas, potentes y asíncronas, útil para la integración de múltiples sistemas. Allí se integran los componentes y facilita la captura de documentos, la conversión de imágenes a texto y la presentación del contraste de información de pedidos de materia prima.

En la quinta fase se lleva a cabo la validación experimental del sistema completo, para ello se realizan 35 pruebas usando información real de una empresa dedicada a la fabricación de productos de aseo. Se calcularon indicadores de coincidencia y diferencia entre los documentos requeridos para cada pedido, así como de similitud de texto con el algoritmo de Ratcliff–Obershelp [18] y de números mediante una función normalizada basada en la

diferencia relativa entre valores. Esta fase permite la identificación de inconsistencias, falsos positivos y limitaciones mediante métricas objetivas del desempeño del sistema diseñado.

La Fig. 1 presenta el diagrama que explica cada una de las fases y actividades de la metodología, mostrando las integraciones de las herramientas utilizadas, lo que facilita replicar el proceso en escenarios similares.

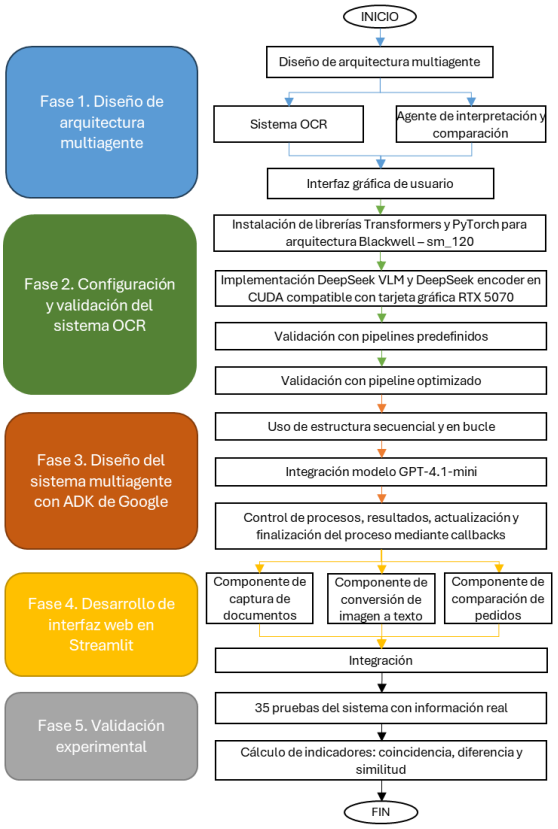

Fig. 1. *Metodología detallada*

3. RESULTADOS

3.1. Arquitectura del sistema multiagente

El problema de la validación cruzada de los documentos se subdivide en cuatro subproblemas computacionales el primero corresponde a la extracción de texto, el segundo a la estandarización y mapeo de texto plano en lenguaje natural hacia esquemas JSON, el tercero a la comparación semántica y validación cruzada lógica entre la información estandarizada, y el cuarto a la comunicación visual y funcional entre el usuario y el sistema. Esto permite seleccionar la tecnología adecuada para cada subproblema. Así, el diseño de la arquitectura del sistema multiagente se estructura, el mismo se presenta en la Fig. 2, donde Deepseek es usado para interpretar los textos en las imágenes de los documentos relacionados con el proceso de compra de materias primas e insumos (pedido, remisión, factura, orden de compra) a analizar, el sistema multiagente se hace usando ADK de Google, que es útil para construir la arquitectura para controlar y monitorear el flujo del proceso de interpretación y comparación, y finalmente se usa Streamlit [19] para la interfaz web que va a permitir la carga de archivos, mostrar los resultados de DeepSeek OCR y del sistema multiagente.

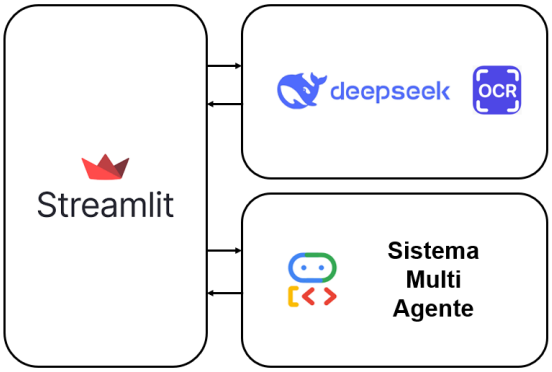

Fig. 2. *Arquitectura del sistema multiagente*

Para el diseño se tuvieron en cuenta cinco criterios. Primero la separación estricta de responsabilidades de percepción y razonamiento para mitigar la degradación del rendimiento, donde Deepseek-OCR se usa exclusivamente para la digitalización textual ya que se encuentra optimizado para interpretar mediante inferencia secuencial en entornos locales con aceleración por hardware (GPU) demandando memoria de video (VRAM), por su parte las capacidades inferenciales se asignaron al modelo de lenguaje GPT-4.1-mini que opera sobre el texto previamente extraído por lo que no requieren capacidades visuales masivas pero si un procesamiento simbólico avanzado para interactuar con esquemas JSON estructurados. El segundo fue la orquestación a través de un pipeline multiagente especializado para dirigir tanto el flujo de datos como de tareas, soportada en ADK que se seleccionó tras compararlos con arquitecturas tradicionales como script monolítico, cadenas lineales simples y agente único centralizado. El tercero fue la implementación de la dupla agente generador y agente validador para evitar errores de omisión de secciones o finalización prematura del proceso, garantizando la validación de la información en la fase de generación. El cuarto corresponde al desacoplamiento a través de un esquema documental paramétrico externo en excel para facilitar el mantenimiento futuro y la

flexibilidad operativa de la solución permitiendo a un analista de procesos modificar variables, tipos de datos y valores por defecto sin compilar el sistema. Y el quinto es el flujo de información entre los agentes usando el estado de sesión de ADK como conducto centralizado de datos eliminando el intercambio directo de mensajes entre agentes.

Los mecanismos de interacción entre los agentes evitan la comunicación directa entre ellos, todo se maneja de forma centralizada con el estado de sesión, el control de los archivos se hace mediante un contador para que el pipeline seleccione los documentos organizados.

3.2. Sistema OCR

Inicialmente se configuro el modelo de DeepSeek [20] en local, posterior a ello se validó su funcionamiento con los pipelines predefinidos. Partiendo de esto se desarrolló un pipeline propio para la inicialización de los modelos (SAM y VLM) que permite optimizar los costos de procesamiento y memoria en cada inferencia.

3.3. Sistema multiagente

Se construyó el sistema multiagente utilizando el concepto de flujo de agentes referente en la documentación del ADK de Google [16], donde se tiene un agente base que se puede extender mediante un LLM, agentes de flujo de trabajo (secuencial, paralelo, en bucle, y/o con lógica de consumo). Basado en esto, se usan los siguientes agentes estructurados:

- Agente interprete ADK: Se le asigna un rol exclusivamente para la transcripción de los datos semánticos, sin hacer suposiciones sobre datos faltantes, relaciona el texto plano del Deepseek-OCR con el archivo excel cargado en tiempo real forzando la salida a JSON válido.
- Agente basado en LLM: utiliza Modelos de Lenguaje Grandes (LLM) como motor principal para comprender el lenguaje natural, razonar, planificar, generar respuestas y decidir dinámicamente cómo proceder o qué herramientas utilizar, lo que los hace ideales para tareas flexibles y centradas en el lenguaje. Se configura con el rol de auditor lógico y tomador de decisiones a partir de la instrucción de verificar correspondencia entre el número de identificación tributaria, tolerancias por redondeo de 0.01 unidades monetarias y consistencia semántica en la descripción de las materias primas, la salida es un informe codificado en JSON con el concepto aprobado o rechazado y una matriz de diferencias.
- Agente Secuencial: Ejecuta sus subagentes en el orden especificado en la lista.
- Agente de Bucle: Ejecuta sus subagentes en un bucle (es decir, de forma iterativa). Ejecuta repetidamente una secuencia de agentes durante un número específico de iteraciones o hasta que se cumple una condición de terminación.

Para el funcionamiento de estos agentes se definió el flujo de proceso ilustrado en la Fig. 3, integrando la salida de Deepseek OCR.

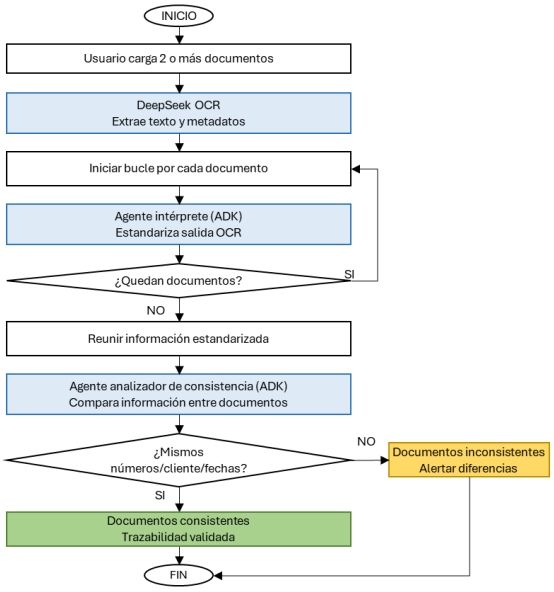

Fig. 3. *Diagrama de flujo del sistema multiagente*

Para el control y monitoreo del proceso, debido a que es un flujo de agentes, se utilizan “Callbacks” como mecanismos de control operativo y sincronización, que en el marco de trabajo existen tres tipos enfocados en las ejecuciones del flujo y cada tipo tiene una opción de control previa y posterior a una ejecución.

Estos métodos permitieron controlar el proceso en los siguientes aspectos:

- Callbacks de Agentes:
  - Contador para monitoreo.
  - Almacenamiento de resultados.
  - Monitoreo de resultados.
- Callbacks de Tools:
  - Control para finalización de iteraciones en el bucle.
- Callbacks de Modelos:
  - Actualización de instrucciones (prompts) dinámica.

Adicionalmente, en este marco se manejan conceptos importantes en el flujo de funcionamiento:

- Sesión: Referente al hilo de conversación actual
  - Representa una interacción única y continua entre un usuario y su sistema de agente.
  - Contiene la secuencia cronológica de mensajes y acciones realizadas por el agente (referidos eventos) durante esa interacción específica.
  - Una sesión también puede contener datos temporales (estados) relevantes sólo durante esta conversación.
- Estados: Datos dentro de la conversación actual
  - Datos almacenados dentro de una sesión.
  - Se utiliza para administrar información relevante únicamente para el hilo de conversación actual activo.

Estos conceptos son útiles ya que tanto los agentes como los "Callbacks" hacen uso de los estados que a su vez se almacenan en la sesión actual de procesamiento, para lo que se definen los siguientes estados con los cuales contará el flujo de agentes para su funcionamiento:

- Cantidad de archivos (contador entero determinista usado como índice de control para la selección de documentos).
- Lista de textos extraídos de cada documento por DeepSeek OCR (colección indexada secuencialmente con las cadenas de texto plano).
- Lista de nombres de archivos procesados (lista de textos).
- Contador de archivos procesados por el flujo de agentes (entero inicialmente 0).
- Lista de textos interpretados por el flujo de agentes (lista de textos que inicialmente es vacía).
- Esquema de interpretación (estructura paramétrica de referencia con los campos mandatorios y tipos de datos requeridos por la organización).

Según el flujo del proceso planteado inicialmente se cuenta con los archivos cargados por el usuario y la salida de texto generada por el modelo DeepSeek, el estado "Esquema de interpretación" es una estructura de datos obtenida a partir de una tabla paramétrica en Excel que permite definir las variables, tipo de variable a extraer e interpretar, y contiene los espacios que se muestran en la Tabla 1.

Esta tabla permite parametrizar y controlar las respuestas del agente interprete que debe tanto clasificar el tipo de documento como generar la salida estructurada predefinida.

Tabla 1: *Tabla predefinida*

Espacio	Descripción
Tipo_documento	Factura, Remisión, Pedido-Orden_Compra
ID_variable	Identificador secuencial
Jerarquía	1=documento, 2=objeto/array template, 3=campo atómico
Variable	Nombre exacto en el JSON de salida
Tipo_variable	string, integer, array, object
Ejemplo	Valor guía para el LLM
Valor_defecto	Valor si el campo no se encuentra en el OCR

Se presenta en la Tabla 2 un ejemplo concreto para la construcción del esquema para un pedido.

Tabla 2: *Ejemplo de tabla predefinida*

Jerarquía	Variable	Tipo	Default
1	nombre_proveedor	string	""
1	materias_primas	array	[]
2	materia_prima	object	{nombre:"", cantidad:0, presentacion:"", precio:0}
3	nombre_materia_prima	string	""
3	cantidad_materia_prima	integer	0
3	presentacion_materia_prima	string	""
3	precio_materia_prima	integer	0

Para garantizar la validación de la información, en formato JSON, en la fase de verificación, el agente validador comprueba que se tenga conformidad sintáctica, se mantenga la integridad del esquema y se cumpla el tipo de datos en cada campo. Si identifica fallas se rechaza la estructura, detalla el error y reinicia el bucle de generación en máximo tres ocasiones para evitar loops infinitos. La Tabla 3 muestra ejemplos representativos de entrada y de salida del proceso. Donde el agente secuencial toma los datos estructurados y los concatena dinámicamente en el prompt del agente interprete delimitando la búsqueda textual, éste último busca patrones que corresponden a las restricciones de la tabla, el agente generador emite la estructura JSON intermedio, que el agente analizador de consistencia

compara, finalmente el agente interprete describe los resultados de la auditoria.

Tabla 3: *Ejemplos de entradas y salidas del proceso*

Tipo	Detalle
Entrada al pipeline (post-OCR)	data = { "file_0": [texto_ocr_pedido, "path/pedido.png"], "file_1": [texto_ocr_factura, "path/factura.png"], } run_pipeline_sync(data, folder_session_id="session_001")
Salida agente intérprete	MATERIA PRIMA: Fragancia Blue Fresh | CANTIDAD: 12 Kg | PRECIO: $64.000
Salida JSON estandarizada	{ "nombre_proveedor": "AROMAS Y PROCESOS SAS", "materias_primas": [{ "nombre_materia_prima": "Fragancia Blue Fresh", "cantidad_materia_prima": 12, "presentacion_materia_prima": "12 Kg", "precio_materia_prima": 64000 }] }
Salida del analizador - Caso exitoso	{ "corresponden": true, "analisis": "Se identificó file_1 como FACTURA. Proveedores ANDLER SAS y HANDLER SAS son equivalentes. Cantidades y productos coinciden.", "documentos_procesados": ["factura", "remision", "pedido"], "documentos_diferentes": [], "diferencias_detectadas": [] }
Salida del analizador - Caso con inconsistencias	{ "corresponden": false, "analisis": "Se comparó factura vs pedido. Diferencias en proveedor y cantidad.", "documentos_procesados": ["factura", "pedido"], "documentos_diferentes": ["file_1_pedido"], "diferencias_detectadas": [{ "documento_comparado": "file_1_pedido", "campo_diferente": "proveedor", "valor_factura": "CONQUIMICA S.A.S.", "valor_comparado": "Industrias Novaquim S.A.S." }, { "documento_comparado": "file_1_pedido", "campo_diferente": "item_detalle", "indice_item": 0, "subcampo": "cantidad", "valor_factura": 200, "valor_comparado": 215 }] }
Salida enriquecida	{ "corresponden": false, "detalle_comparacion_metricas": { "file_1_pedido":[{"campo": "proveedor", "similitud": 0.372, "similitud_numerica": null}, {"campo": "item_detalle", "subcampo": "cantidad", "similitud": 0.333, "similitud_numerica": 0.930}] } }

3.4. Interfaz web de interacción

Se desarrolla una aplicación web para interactuar con el sistema de interpretación y comparación de archivos, en la Fig. 4 se observa la pantalla inicial donde se suben las imágenes de los documentos asociados a cada compra y se envían al sistema.

Fig. 4. *Pantalla inicial del sistema multiagente*

La Fig. 5 muestra una vista previa de la carga de documentos (2 archivos) y la ejecución de DeepSeek OCR del sistema donde cada recuadro de color demarca los campos a comparar.

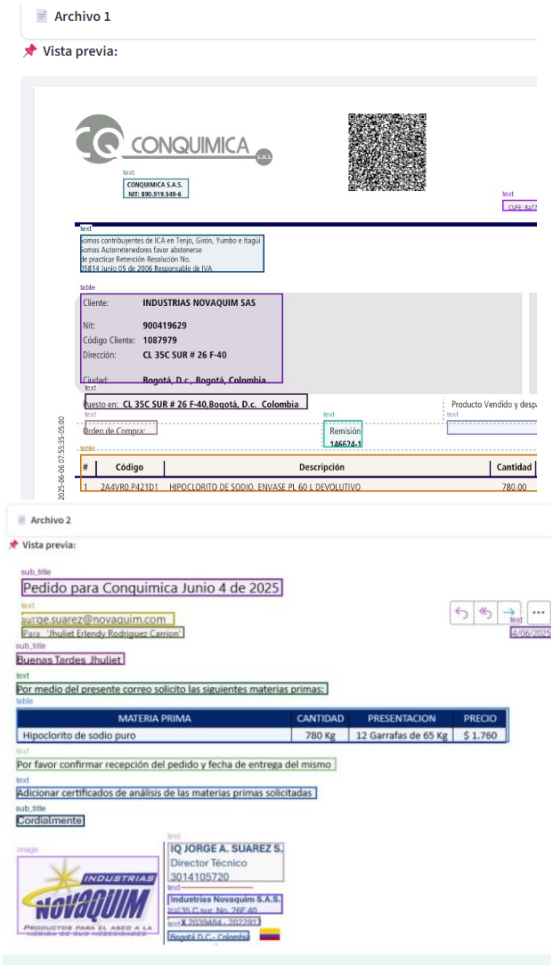

Fig. 5. *Pantalla carga de documentos y ejecución de DeepSeek OCR del sistema multiagente*

Finalmente, para este caso de ejemplo la respuesta generada por el sistema multiagente es *"Solo hay dos documentos para comparar: la factura (file_0) y el pedido (file_1). El proveedor en la factura es CONQUIMICA S.A.S., mientras que en el pedido aparece como Industrias Novaquim S.A.S., que es el cliente, por lo que se corrigió la inversión. Se detecta diferencia en el proveedor tras la corrección. Además, en los items, la cantidad de ácido sulfónico lineal difiere: 200 en la factura y 215 en el pedido, y el precio unitario también difiere (8263 vs 9038). El alcohol etílico coincide en cantidad y precio. No hay remisión para comparar".*

3.5. Validación experimental

Para la validación experimental, se usaron datos reales de una empresa dedicada a la fabricación y comercialización de productos líquidos de aseo. Se ejecutaron 35 pruebas en total, de las cuales una no contenía documentos y 34 pruebas en donde se cargaron entre 1 y 3 archivos referentes por cada caso (facturas, pedidos, órdenes de compra y/o remisiones), 28 pruebas con documentos se ejecutaron correctamente, lo que corresponde al 82,353% de éxito, los fallos se presentaron en la respuesta generada por el "agente analizador de consistencia" ya que no eran respuestas estructuradas o no cumplían con los parámetros establecidos en las instrucciones.

De acuerdo con la información disponible en la empresa, la distribución de la cantidad de documentos estudiados en cada caso, en las pruebas exitosas, se pueden observar en la Fig. 6. Se hizo una prueba con un solo documento (3.57%) para verificar que la identificación de la correspondencia fuera falsa, resultado que coincidió con lo esperado, se ejecutaron 11 pruebas (39.29%) comparando 2 documentos y 16 (57,14%) comparando 3 archivos.

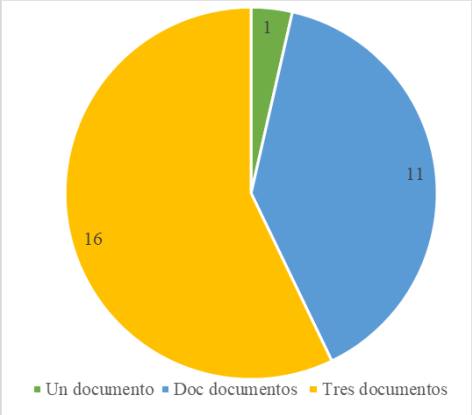

Fig. 6. *Cantidad de documentos comparados en las pruebas*

El uso de métricas de similitud, tanto textual como numérica, permite obtener un análisis más preciso de las diferencias que se detectan, lo que brinda información útil y en tiempo real para los responsables de la gestión de pedidos de materias primas.

La comparación de la información en los documentos se hace para los campos: proveedor, cantidad, unidad de medida, descripción del producto y precio unitario. Por ello, se calculan métricas de similitud con base en los valores cualitativos (proveedor y producto) y cuantitativos (precio, cantidad) con el fin de afinar el análisis del sistema, usando el algoritmo de Ratcliff–Obershelp para texto (1) y para cantidad (2), respectivamente.

$$Similitud(A,B) = \frac{2*M}{|A+B|} \quad (1)$$

Donde M= número de caracteres coincidentes en los substrings más largos comunes, A = longitud de A, y B = longitud de B.

$$Similitud(A,B) = 1 - \frac{A-B}{\max(A,B)} \quad (2)$$

Donde A = valor de A y B = valor de B

Dichas métricas solo se calculan cuando el sistema reporta inconsistencias, lo que permite detectar en cuál de los campos (proveedor, cantidad, unidad de medida, descripción del producto y precio unitario) existen diferencias para informar de forma oportuna a los responsables de la gestión de compras de materias primas. En la Fig. 7 se pueden ver los resultados de estas métricas aplicadas por el sistema en las pruebas a nivel general, se observan similitudes numéricas con valores extremos que corresponden a poca o mucha coincidencia, la similitud verbal si tiene un comportamiento más uniforme, pero con mayor concentración en datos bajos que corresponden a baja similitud.

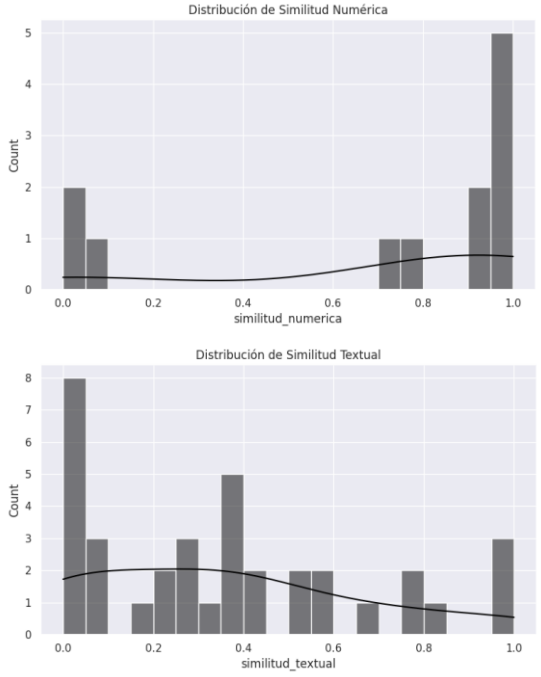

Fig. 7. *Resultado de pruebas de similitud numérica y textual*

Se evidenció que aun cuando el sistema encuentra campos vacíos realiza comparaciones por lo que reporta valores de cero en la similitud, también

arroja valores de similitud de 1 que corresponde a campos idénticos, lo que hace necesario revisar la parametrización definida para las interpretaciones en estos dos puntos extremos.

Para la evaluación de desempeño del prototipo se diseñó una validación experimental orientada a medir dos capacidades principales del sistema: la calidad de extracción de información desde documentos de distinto tipo y la capacidad del agente analizador para identificar consistencia o inconsistencia documental respecto a un criterio de referencia.

La validación se realizó sobre 35 casos, que en conjunto agruparon 89 documentos procesados (entre facturas, pedidos y remisiones) mediante los siguientes métodos de análisis:

- Evaluación de extracción documental: Se comparó la información extraída automáticamente contra el esquema de variables, calculando el porcentaje de completitud (3) para cada caso entre el cociente de los campos extraídos correctamente (CEC) sobre los Campos esperados (según esquema) (CE). Esta métrica permite cuantificar la capacidad para extraer información independientemente del resultado final.

$$Completitud = \frac{CEC}{CE} * 100 \qquad (3)$$

- Evaluación del agente analizador: Posteriormente, el agente de análisis recibió únicamente la salida estructurada del pipeline y generó un veredicto binario de Consistente (True): No se identifican diferencias relevantes entre documentos. Inconsistente (False): Existen diferencias que requieren revisión. Sin veredicto: El sistema no logró emitir una salida válida o estructurada.

La evaluación permitió evidenciar que el sistema alcanzó una completitud promedio de extracción del 78,3 %, con una mediana de 81,3 %, indicando que la mayoría de los casos logró recuperar más del 80 % de la información esperada. Los resultados se resumen en la Tabla 4.

Tabla 4: *Resultados de validación*

Métrica	Resultado
Casos evaluados	35
Documentos procesados	89
Completitud promedio	78,3 %
Mediana de completitud	81,3 %
Veredictos de experto disponibles	34

A partir del comportamiento del sistema se puede definir que el prototipo presenta una interpretación documental estable y confiable, aunque persisten pérdidas de información asociadas a variabilidad estructural entre documentos y documentos con baja calidad.

Adicionalmente, se realizó un análisis por cada tipo de documento para identificar el impacto de cada uno en el desempeño del sistema, cuyos resultados se relacionan en la Tabla 5.

Tabla 5: *Resultados por tipo de documento*

Tipo	Factura	Pedido	Remisión
Documentos	35	36	18
Extraídos	299	424	80
Esperados	389	501	111
Completitud global	76,9 %	84,6 %	72,1 %
Promedio por caso	74,3 %	84,6 %	72,8 %

Los resultados muestran que el tipo de documento *Pedido* fue el documento con mejor extracción, alcanzando una completitud del 84,6%, mientras que *Remisión* presentó el menor desempeño (72,1%), debido a una mayor complejidad o menor estandarización del formato. Por otro lado, cuando se evaluó el comportamiento del agente analizador (sin tener en cuenta la referencia manual) se obtuvo la distribución de decisiones que se tabulan en la Tabla 6.

Tabla 6: *Distribución de veredictos*

Veredicto	Casos	Participación
Inconsistente (False)	20	57,1 %
Consistente (True)	9	25,7 %
Sin veredicto	6	17,1 %

Se observa una tendencia hacia la detección de inconsistencias, comportamiento deseable en escenarios de recepción y validación documental.

Para medir la confiabilidad del sistema se comparó la sugerencia generada automáticamente del sistema contra la revisión manual realizada para cada uno de los casos. Se definió como clase positiva la detección de inconsistencia documental (False) lo que permitió elaborar la matriz de la Tabla 7.

Tabla 7: *Matriz de resultados*

	Experto: Inconsistente	Experto: Consistente
Agente: Inconsistente	16	4
Agente: Consistente	4	4

A partir de esta matriz se calcularon las métricas de clasificación relacionadas en la Tabla 8.

Tabla 8: *Métricas de clasificación*

Métrica	Valor
Casos evaluables	28
Casos no evaluables	7
Precisión	0,800
Exactitud	0,714

Los resultados permiten observar que el sistema logra un equilibrio entre la capacidad de detección y estabilidad, definiendo una respuesta consistente tanto para identificar inconsistencias reales como para limitar casos falsos. Sin embargo, la exactitud del 71,4 % muestra que aún existe margen de mejora asociado a errores de extracción que afectan etapas posteriores del análisis y definición del sistema, casos en donde el agente no logra generar una salida válida.

Los factores que favorecen el éxito son contar con un OCR de calidad, que el proveedor sea claramente identificable en la factura, la correspondencia entre códigos de referencia en pedido y factura y contar con el conjunto completo de documentos del mismo proceso. Los casos sin éxito se dan por las fallas en la interpretación y no en el análisis, dado que los campos que no entiende quedan como null. La principal causa de los errores es la confusión entre el proveedor y el cliente, pese a que el agente analizador detecta la inversión se limita a reportar el error, pero no lo corrige.

Este procedimiento de comparación de documentos actualmente es ejecutado de forma manual por un colaborador de la empresa, actividad en la que gasta cerca de 7 minutos al comparar 2 documentos (factura contra orden de compra), y aproximadamente 10 minutos si incluye remisión, en varias ocasiones la comparación no se hace de manera precisa como consecuencia de los errores humanos, en contraste el tiempo que toma el sistema multiagente para validación de pedidos de materia prima promedio es inferior a 1 minuto (57.2 segundos) y a 2 minutos (80.85 segundos) para 2 y 3 documentos, respectivamente, aportando también a la eliminación de errores humanos. Vale aclarar que el código no tiene timers, estos tiempos se calcularon a partir de las marcas temporales de las pruebas realizadas en un equipo con GPU NVIDIA GeForce RTX 5070 (12 GB VRAM), CPU AMD Ryzen 9 9950X de 16 núcleos, con RAM de 16 GB (WSL2), Sistema operativo Windows 11 + WSL2 y Driver NVIDIA 591.86.

Las limitaciones prácticas del sistema están asociadas a la dependencia de hardware especializado, la dependencia de servicios externos, la reproducibilidad y la escabilidad. En primer lugar, el funcionamiento secuencial del OCR no es ejecutable en una CPU sino que requiere una GPU con más de 16 GB de VRAM. En segundo lugar, el LLM (GPT-4.1-mini) se conecta por API lo que hace que la latencia dependa de la red y la disponibilidad de la API, se usa una temperatura de cero para reducir variabilidad, pero el servicio puede actualizar el modelo, de manera alternativa se podría usar Gemini 2.0 Flash (no evaluado). Para la reproducibilidad solo se cuentan con casos sin anotaciones manuales lo que no permite calcular métricas de uso generalizado para este tipo de aplicaciones por lo que se propone hacer las anotaciones correspondientes y dado que la cantidad máxima de iteraciones es 3, podría ampliarse de manera proporcional si se requieren más documentos. En cuanto a la escalabilidad el OCR es secuencial lo que hace que el procesamiento de lotes grandes de documentos sea lento.

4. CONCLUSIONES

Las fases de la metodología fueron determinantes para el funcionamiento del sistema multiagente. Las primeras dos fases permiten establecer un pipeline de OCR optimizado para reducir el uso de memoria y de requisitos computacionales, gracias al procesamiento de tokens visuales con SAM de Meta y el decodificador de DeepSeek, asegurando la captura de datos eficiente. En la fase de diseño del sistema, la utilización de agentes secuenciales y en bucle, permitió dividir la validación en tareas de clasificación, interpretación y comparación, usando el modelo GPT-4.1-mini para lograr una interpretación coherente. El desarrollo de la interfaz web facilita la carga de documentos y la visualización de reportes, haciendo el sistema amigable para los responsables de compras. Finalmente, la validación experimental permite la identificación de fallos, el cálculo de métricas de similitud y la detección de limitaciones, además permite pasar de tener un diseño teórico a contar con una herramienta validada con datos reales.

El diseño, implementación y verificación del sistema multiagente para validación de pedidos de materia prima usando una arquitectura que integra el modelo DeepSeek OCR con un sistema Multi Agente basado en el marco de trabajo ADK y una interfaz web creada en Streamlit, permitió la ejecución exitosa del flujo de interpretación y

comparación de información en el 82,353% de las pruebas realizadas, evidenciando una capacidad funcional en escenarios reales de operación para el análisis de documentos heterogéneos relacionados con el proceso de compra de materias primas, como facturas, órdenes de compra, pedidos y remisiones. La tabla paramétrica en Excel se constituye como una ventaja importante al funcionar como un esquema dinámico que puede ser modificado por usuarios no expertos al contar con 3 inyección vía callback. La separación de OCR y del razonamiento permite integrar la operación local del OCR con la operación de agentes vía API, por lo que se cuenta con capas evaluables independientemente. Finalmente, el cálculo de métricas cuantitativas de similitud textual y numérica por campo se validó con mediciones en analítica de campos.

Se identificaron dos tipos de limitaciones, la primera relacionada con la generación de respuestas no estructuradas en el 17.647% de los casos, dada la falta de esquemas de salida definidos, y la segunda asociada a la presencia de valores extremos (0 y 1) en los resultados de las métricas de similitud, que indican la necesidad de ajustar el sistema. El cuello de botella es la interpretación no el análisis, los fallos en el pipeline surgen por la mala clasificación o por la extracción incompleta del agente interprete, que funciona perfecto cuando los JSON son válidos.

Para optimizar la precisión del sistema y mitigar las inconsistencias detectadas, es necesario realizar un ajuste técnico en la parametrización y las instrucciones (prompts) de los agentes, enfocándose prioritariamente en reglas de desambiguación para que el "agente interpretador" distinga correctamente entre proveedor y cliente, eliminando así las comparaciones erróneas contra valores nulos. Adicionalmente, se requiere reforzar la estructura de salida del "agente analizador de consistencia" para garantizar el cumplimiento estricto de los parámetros JSON establecidos y evitar fallos de ejecución, al tiempo que se deben recalibrar los umbrales de aceptación del algoritmo de Ratcliff-Obershelp para las variables textuales, con el fin de mejorar la identificación de diferencias cualitativas, lo que se plantea como trabajo futuro.

AGRADECIMIENTOS

Los autores agradecen a la Universidad Militar Nueva Granada, donde son profesores de planta de tiempo completo. Producto derivado del proyecto de investigación titulado “Fortalecimiento de los procesos de recepción de pedidos y control de inventario de materias primas con el apoyo de la Industria 4.0” INV-ING-4150 financiado por la vicerrectoría de investigaciones de la Universidad Militar Nueva Granada, vigencia 2025.

REFERENCIAS

[1] W. Chu, S. Yin, L. Huang, L. Lin, X. Wang, Z. Zhang and H. Li, "Verify-Agent: Large Language Model Multi-Agent for Intelligent Verification," in *2024 International Conference on Ubiquitous Computing and Communications (IUCC)*, Chengdu, China, 2024, doi:10.1109/IUCC65928.2024.00072.

[2] P. H. Luzolo, Z. Elrawashdeh, I. Tchappi, S. Galland and F. Outay, "Combining multi-agent systems and artificial intelligence of things: Technical challenges and gains," *Internet of Things*, vol. 28, p. 101364, 2024, doi:10.1016/j.iot.2024.101364.

[3] M. Z. R. Farazi, "Enhancing supply chain resilience with multi-agent systems and machine learning: a framework for adaptive decision-making," *The American Journal of Engineering and Technology*, vol. 7, no. 3, pp. 6-20, 2025, doi:10.37547/tajet/Volume07Issue03-02.

[4] M. Mousa, D. Van de Berg, N. Kotecha, E. A. del Rio Chanona and M. Mowbray, "An analysis of multi-agent reinforcement learning for decentralized inventory control systems," *Computers & Chemical Engineering*, vol. 188, p. 108783, 2024, doi:10.1016/j.compchemeng.2024.108783.

[5] G. Ziegner, M. Choi, H. M. C. Le, S. Sakhuja and A. Sarmadi, "Iterative Multi-Agent Reinforcement Learning: A Novel Approach Toward Real-World Multi-Echelon Inventory Optimization," *arXiv*, 2025, doi:10.48550/arXiv.2503.18201.

[6] N. Kotecha and A. del Rio Chanona, "Leveraging graph neural networks and multi-agent reinforcement learning for inventory control in supply chains," *Computers & Chemical Engineering*, vol. 199, p. 109111, 2025, do:10.1016/j.compchemeng.2025.109111.

[7] J. Hu, L. Xia, T. Huang and H. Wu, "A multi-agent deep reinforcement learning approach for multi-echelon inventory optimization and its application to the beer game," *Transportation Research Part E: Logistics and Transportation Review*, vol.

203, p. 104367, 2025, doi:10.1016/j.tre.2025.104367.

[8] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han and S. Park, "OCR-Free Document Understanding Transformer," in *Computer Vision – ECCV 2022. ECCV 2022. Lecture Notes in Computer Science*, vol. 13688, S. Avidan, G. Brostow, M. Cissé, G. Farinella and T. Hassner, Eds., Springer, 2022, doi:10.1007/978-3-031-19815-1_29.

[9] S. Chen, X. Guo, Y. Li, T. Zhang, M. Lin, D. Kuang, Y. Zhang, Z. Zhou and W. Chen, "Ocean-OCR: Towards general OCR application via a vision-language model," *arXiv*, 2025, doi:10.48550/arXiv.2501.15558.

[10] L. Hamdi, A. Tamasna, P. Boisson and T. Paquet, "VISTA-OCR: Towards generative and interactive end to end OCR models," *arXiv*, 2025, doi:10.48550/arXiv.2504.03621.

[11] G. Monteiro, L. Camelo, G. Aquino, R. Fernandes, R. Gomes, A. Printes, I. Torné, H. Silva, J. Oliveira and C. Figueiredo, "A Comprehensive Framework for Industrial Sticker Information Recognition Using Advanced OCR and Object Detection Techniques," *Applied Sciences*, vol. 13, no. 12, p. 7320, 2023, doi:10.3390/app13127320.

[12] R. Sapkota and M. Karkee, "Object detection with multimodal large vision-language models: An in-depth review," *Information Fusion*, vol. 126, no. Part A, p. 103575, 2026, doi:10.1016/j.inffus.2025.103575.

[13] K. Medini, D. Romero and T. Wuest, "Developing a multi-agent system to support multi-variant production ramp-up management," *Smart and Sustainable Manufacturing Systems*, vol. 5, no. 1, pp. 129-147, 2021, doi:10.1520/SSMS20200082.

[14] Z. Zhao, D. Tang, C. Liu, L. Wang, Z. Zhang, H. Zhu, K. Chen, Q. Nie and Y. Ji, "A Large language model-based multi-agent manufacturing system for intelligent shopfloors," *Advanced Engineering Informatics*, vol. 69, no. Part A, p. 103888, 2026, doi: 10.1016/j.aei.2025.103888.

[15] V. Jannelli, S. Schöpf, M. Bickel, T. Netland and A. Brintrup, "Agentic LLMs in the supply chain: towards autonomous multi-agent consensus-seeking," *International Journal of Production Research*, p. 1–31, 2025, doi:10.1080/00207543.2025.2604311.

[16] Google, "Agent Development Kit," 2025. [Online]. Available: https://google.github.io/adk-docs/agents/. [Accessed 28 Enero 2026].

[17] A. Bandi, B. Kongari, R. Naguru, S. Pasnoor and S. Vilipala, "The Rise of Agentic AI: A Review of Definitions, Frameworks, Architectures, Applications, Evaluation Metrics, and Challenges," *Future Internet*, vol. 17, no. 9, p. 404, 2025, doi:10.3390/fi17090404.

[18] H. Guo, Y. Li, Y. Liu and W. Li, "Evaluation of string comparators for record linkage in Chinese environment," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 20, no. 6, p. 2250023, 2022, doi:10.1142/S0219691322500230.

[19] C. Nantasenamat, A. Biswas, J. Nápoles-Duarte, M. I. Parker and R. L. Dunbrack, "Chapter 27 - Building bioinformatics web applications with Streamlit," in *Cheminformatics, QSAR and Machine Learning Applications for Novel Drug Development*, K. Roy, Ed., Academic Press, 2023, pp. 679-699, doi:10.1016/B978-0-443-18638-7.00001-3.

[20] H. Wei, Y. Sun and Y. Li, "Deepseek-OCR: Contexts optical compression.," *arXiv* , 2025. https://doi.org/10.48550/arXiv.2510.18234.